李平 编著
Li Ping

LaTeX 2ε 及常用宏包使用指南

A Guide to LaTeX2e and Commonly Used Packages

李平 编著

Li Ping

LaTeX2e 及常用宏包使用指南

A Guide to LaTeX2e and Commonly Used Packages

清华大学出版社

内 容 简 介

LaTeX是以TeX为引擎的高质量格式化排版系统，它容易学习而且支持命令宏，具有很好的可扩展性。LaTeX2e是LaTeX的新标准，它使用户能更有效地使用LaTeX及各种扩展宏包。

本书通过大量的实例由浅入深、由易而难地介绍了LaTeX2e排版系统的基础知识以及一些经常使用的LaTeX2e扩展宏包，着重叙述了设计版面、制作表格、排版数学公式、绘制简单图形、插图处理以及排版中文稿件等方面的内容。本书可作为LaTeX2e的入门书籍，也可作为继续深入了解LaTeX2e知识的参考书。

本书适合所有需要科技文献排版的读者，如理工科大学的学生和教师、科技工作者、编辑等。

图书在版编目(CIP)数据

LaTeX2e及常用宏包使用指南/李平编著. —北京：清华大学出版社，2004

ISBN 7-302-08147-6

Ⅰ. L… Ⅱ. 李… Ⅲ. 排版—应用软件，LaTeX2e Ⅳ. TS803.23

中国版本图书馆CIP数据核字(2004)第013403号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社 总 机：010-62770175　　**客户服务**：010-62776969

责任编辑：王海燕

封面设计：常雪影

印 刷 者：北京四季青印刷厂

装 订 者：三河市新茂装订有限公司

发 行 者：新华书店总店北京发行所

开　　本：185×230　**印张**：17　**字数**：348千字

版　　次：2004年4月第1版　2004年4月第1次印刷

书　　号：ISBN 7-302-08147-6/TP·6062

印　　数：1～3000

定　　价：29.00元

前　言

TeX 是一个格式化排版系统，它一问世便以其排版效果的高质量震动整个出版界。尤其是在排版含有大量数学公式的科技文献方面更显示了它的优越性。TeX 还是一个源代码公开的免费排版系统，因此吸引了许多计算机专家及 TeX 爱好者为之添砖加瓦。LaTeX 就是一个以 TeX 为引擎的功能强大且使用方便的排版系统。随着计算机技术的发展和计算机的普及，LaTeX 的功能以迅猛的速度进一步发展和壮大，出现了许多扩展 LaTeX 功能的优秀宏包。为了更方便、更有效地使用 LaTeX 及各种宏包，一个由 TeX 专家组成的 LaTeX3 小组发布了 LaTeX2e。现在国际上大多学术部门和校园网上都安装有 TeX/LaTeX2e 系统，许多出版机构都采用 LaTeX2e 来排版书刊，不少出版社还要求作者提供手稿的 LaTeX2e 源文件。我国已经有不少大专院校的教师和学生、研究所的科研人员及出版社的编辑在使用 TeX/LaTeX，也有部分出版社开始接受用 LaTeX 排版的稿件。虽然 TeX/LaTeX 不是一天就能掌握的排版系统，但是一旦开始使用就会被它无穷的魅力所吸引。随着科技的进步、网络时代的到来，一定会有越来越多的人喜欢用 LaTeX 排版自己的文稿。

本书详细介绍了 LaTeX2e 排版系统的基础知识以及一些经常使用的 LaTeX2e 扩展宏包；着重叙述了在掌握了一定的排版知识后如何修改排版效果的知识；介绍了如何制作表格、如何排版数学公式；对 LaTeX2e 的图形功能（包括作图和插图）做了大量阐述；也介绍了如何安装 CJK 宏包并用它来排版中文稿件。书中包含大量的例子，大多数例子中的排版结果置于右边阴影小页中，而源代码则放在左边，这使读者便于阅读和理解。因为本书就是用 LaTeX2e 配合 CJK 宏包以及所介绍的扩展宏包写成的，所以大部分阴影小页中的排版结果都是由左边的源代码自动生成的。本书目录和索引都很详细便于查阅。

全书分 9 章：第 1 章简要介绍了 TeX 和 LaTeX 的历史。第 2 章介绍了有关 LaTeX 源文件、命令及环境的基础知识，并且对盒子、计数器、控制结构语句等重要专题也做了详细的介绍。第 3 章介绍了有关文稿的章节等逻辑结构的排版，并介绍了修改和制定页版式的 titlesec 和 fancyhdr 宏包及排版目录的 titletoc 宏包。第 4 章介绍了如何制作表格。第 5 章介绍了如何用 AMSLaTeX 宏包套件来排版数学公式。第 6 章介绍了 LaTeX 基本作图环境及一些扩展作图宏包。第 7 章详细介绍了如何插入图片以及如何对插入的图片做必要的排版处理。第 8 章介绍了处理浮动体的知识。第 9 章介绍了如何安装和使用排版中文稿件的 CJK 宏包。

由于作者水平有限，书中错误在所难免，欢迎广大读者批评指正。

编　者

目　录

第 1 章　LaTeX 历史简介

随着计算机技术的发展和计算机的普及，利用电子排版系统在家中排版出精美的书稿和论文已不是一件难事。目前市场上有许多包含各种功能的电子排版软件，这些排版软件基本上可以分为两类：第一类是所谓具有“所见即所得”功能的文字处理软件，例如 Microsoft Word、WPS 等。这类软件都有功能丰富的菜单，并且所编辑的文件呈现在屏幕上的式样也就是打印出来的文稿的式样。第二类称为“格式化排版程序”。它基于两个步骤，首先编辑一个文本文件，称为“源文件”，然后用排版程序对源文件进行处理并将结果传到输出设备，如打印机或高清晰度的屏幕。如果对输出结果不满意，就需要修改源文件，然后再用排版程序对源文件进行处理和输出。

这两类排版系统各有特点，也各有自己的适用范围。在排版高质量的书籍或论文，特别是含有大量数学公式的科技文献方面，用第二类的格式化排版程序就比较合适。另外，在利用一个固定的文本文件获得不同的排版结果方面，第二类的排版程序也特别有用。例如，在源文件中插入一个命令，就可以使原本单列排版的文章变为多列排版的。

1.1　TeX 和 LaTeX

TeX 就是一个功能强大的特别适合排版科技文献和书籍的格式化排版程序。它是由著名计算机专家和数学家美国斯坦福大学 D. E. Knuth 教授研制的。20 世纪 60 年代，Knuth 准备出系列专著《计算机程序设计技巧》(The Arts of Computer Programming)。前三册已经出版，当他正在撰写第四册时，出版社拿来第二册的第二版给他过目，结果令他大失所望，因为当时出版社的印刷技术没有使他的书稿更好看，反而变糟了，尤其是在数学公式和字体上面的缺陷更令他无法接受。于是他就打算自己写一个既能供科学家编排手稿又符合出版社印刷要求的高质量的计算机排版系统。这就是 TeX 排版系统的由来。TeX 的名称来源于希腊字母 $\tau\epsilon\chi$, 因此最后一个字母的发音为苏格兰语中的“ch”，或德语中的“ach”（汉语中读“克”），而不是英文字母 x 的发音。Knuth 为这个系统的名称设计了专用的形式：TEX，它可用命令 \TeX 排印出来，为了方便的缘故，也常写成“TeX”。

Knuth 于 1977 年开始构造 TeX 系统，1982 年成功开发初版，之后又有几次升级。他用无理数 π 的近似值作为此系统的版本号，每升级一次其版本号就增加一位数字。TeX 系统内核相当稳定，1995 年以后其版本号一直停止在 3.14159，直到最近才又进行了一次升级。目前 TeX 系统的版本号是 3.141592，几乎没有“bug”。在 Knuth 开始构造 TeX 系统的年代，用于科技排版的字库都非常昂贵，因此伴随着 TeX 系

统，Knuth 还开发了一个用 Bézier 曲线描述字符边界的造字系统 METAFONT，标准的 TeX 系统所包含的所有字符均可由 METAFONT 构造出来。

TeX 系统是由 Pascal 语言编写的，程序的源代码也是公开的。它包含 300 条基本命令和 600 条扩展命令，几乎可以排版任何格式的文献，如一般文章、报告、书刊和诗集等，对数学公式的排版也被公认为是最好的。TeX 系统的优点之一是它还支持命令宏，这使得使用 TeX 成为一种乐趣，用户可以自己编写宏包来定义更多、更方便的新命令，这也是 TeX 能得以迅速发展的原因。而且 TeX 是一个可移植性系统，可以运行于所有类型的计算机（如苹果机、 IBM PC 机及大型工作站）和各种操作系统（如 DOS、Windows、Unix 等），它的排版结果 dvi 文件与输出设备无关，可以在不同的操作系统上显示和打印。TeX 源文件是 ASCII 码文件，可以方便地在网络上传输。目前，大多数学术部门和校园网上都安装有 TeX 系统。国际上许多出版机构也采用 TeX 系统来排版书刊，不少出版社还要求作者提供手稿的 TeX 源文件。

虽然 TeX 的功能非常强大，用它可以排版任何式样的文稿，但普通用户要灵活掌握 TeX 的 900 条初始命令还是有困难的。因而，TeX 公开几年后，利用 TeX 的宏定义功能开发的宏库 AMSTeX 和 LaTeX 就产生了。AMSTeX 是 Michael Spivak 受美国数学会 (AMS) 的委托编写的，主要用于 AMS 和其分支机构出版的大量书籍、期刊和评论。AMSTeX 含有一个宏包 (Style file)，供作者用来方便地准备稿件。用 AMSTeX 可以方便地排印出非常复杂的数学公式和 AMS 制定的全部数学符号。

LaTeX 是由美国计算机学家 Leslie Lamport 于 1985 年开发成功的。尽管在排版数学公式和数学符号方面 LaTeX 不如 AMSTeX，但 LaTeX 提供了大量易于学习和使用的命令，如非常有用的交叉引用命令 (cross-referencing commands)，这是 AMSTeX 所不具备的。因而 LaTeX 有更广泛的用途，特别是在排版信件、书刊、诗集等方面更优于 AMSTeX。TeX 的命令好比是建筑所使用的各种各样的材料，优秀的建筑师用它能建造出各种美丽的建筑； LaTeX 的命令好比是已经建造好的各种各样的房间和家具，用户只需选择适合自己的房间和家具就能得到满意的住所，而且这种房间和家具之多几乎无须用户自己动手建造。为了使用户既能使用 LaTeX 提供的大量命令，又能排版出优美的数学公式和数学符号，美国数学会又开发了 AMSLaTeX。

1.2 LaTeX2e

由于 LaTeX 的诸多优点，许多 LaTeX 爱好者都为它撰写了不同作用的宏库或宏包，存放在 Internet 的不同地方，但有些宏包的作用大同小异，命令相互冲突互不兼容，还有些宏包看起来不太清楚是为哪种格式的 TeX 编写的。为了结束这种令人不满

意的状态，把各种宏包统一在一种格式之下，由 Frank Mittelbach 领导成立了 LaTeX 3 小组，其主旨是重写 LaTeX 系统，构造一个有效的优化内核，把一些常用的宏包提供的具有较好特殊功能的命令包含进去，使新系统的功能更完备、更强大，并且更易于使用。因为建造 LaTeX 3 是一项长期的工作，所以为了目前的使用，LaTeX 3 小组公布了介于 LaTeX 2.09 和 LaTeX 3 之间的版本，称为 LaTeX 2_ε，可以用命令 `\LaTeXe` 排印此名称，为便于书写，也常写成 LaTeX2e。在这个新版本中，由 AMSLaTeX 提供的新字体选择方案 (New Font Selection Scheme) 成为标准的字体选择法。以往的一些宏库如 AMSLaTeX 和 SliTeX 等都成为 LaTeX2e 的扩展宏包套件，只要在 LaTeX2e 源文件中用 `\usepackage` 命令调用这些套件中的宏包就可以使用它们了。区别于旧版 LaTeX 2.09，在 LaTeX2e 中源文件的第一条命令已由 `\documentstyle` 改为 `\documentclass`。这就使 LaTeX 程序能自动区分新旧格式的源文件，从而切换到与之相兼容的模式中。本书讨论的就是如何使用 LaTeX2e 及一些常用的宏包。

1.3 文稿产生过程

LaTeX 源文件是 ASCII 码的文本文件。它包含文件的文本部分及通知 LaTeX 程序如何编排这些文本的命令部分。LaTeX 源文件的完整名称应包含两个部分：文件名和扩展名 `.tex`（例如 `myart.tex`）。文件的名称有时受计算机系统的限制，例如，文件名的字符个数不能超过 8 个。此时，`internal.tex` 是有效的名称，而 `international.tex` 却不是。

要得到排版结果必须运行 LaTeX 程序来编译源文件。如何运行 LaTeX 程序要视 LaTeX 系统是如何安装的。通常是键入命令 `latex`，紧跟着输入不含扩展名的文件名。例如，要编译文件 `myart.tex`，只要键入

```
latex myart
```

即可。LaTeX 程序运行时，屏幕上会出现一些提示和出错信息，根据这些信息可对源文件进行修改。源文件经过编译后，系统会自动生成三个文件：以 `log` 为扩展名的 `log` 文件、以 `aux` 为扩展名的 `aux` 文件和以 `dvi` 为扩展名的 `dvi` 文件。`log` 文件中记录了编译源文件时产生的提示和出错信息，`aux` 文件中记录了有关交叉引用等辅助信息，`dvi` (device independent) 文件就是 LaTeX 系统按照源文件中的命令对文本进行排版之后形成的格式化文件，它是与输出设备（如屏幕和打印机）无关的。最后还得调用打印机的驱动程序将 `dvi` 文件打印成文稿，或调用屏幕预览程序将 `dvi` 文件输出到屏幕进行浏览。由于 `dvi` 文件中包含了字库信息，在另一台计算机的 TeX 系统中也许不能正确显示和打印。不过用户可以利用 dvips 程序先将 `dvi` 文件转换成 PS（Postscript）

文件。PS 文件是一个图形文件，任何有 Postscript 功能的打印机都能将之直接打印出来。如果打印机没有 Postscript 功能，可以安装 gsview 程序来浏览和打印 PS 文件。另外，用 dvipdfm 程序可以把 dvi 文件转成流行的 PDF 格式文件。也可用 pdflatex 程序直接把 LaTeX 源文件编译成 PDF 文件。图 1.1 是从编辑 LaTeX 源文件开始到得到各种输出结果的一个流程图。

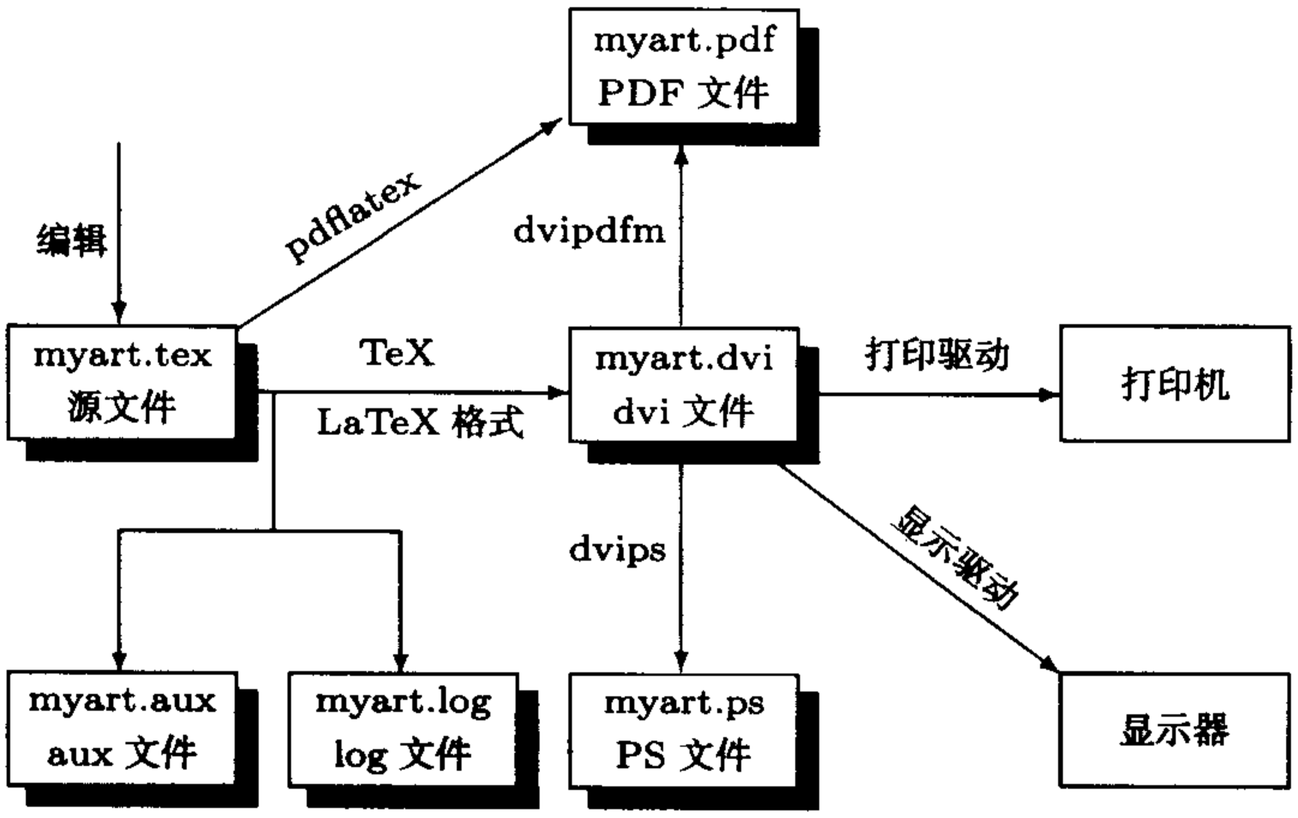


图 1.1 使用 LaTeX

1.4 TeX 和 LaTeX 网络资源

由于 TeX 和 LaTeX 是公开的高质量排版系统，又由于它良好的兼容性和可扩展性以及安装和使用的方便性，近年来 TeX 和 LaTeX 的发展极其迅速，用户已遍布全世界。还成立了许多 TeX 用户组织 (TeX User Groups，简称 TUG)，出版了相关的杂志 (TUGBoat)，专门研讨 TeX 的发展和使用的技术问题。现在 Internet 网络上存有大量与 TeX 和 LaTeX 相关的免费资源供用户自由下载，发展起来几个“综合 TeX 资源网” (Comprehensive TeX Archive Network)，简称 CTAN，以及遍布各国的镜像网站。有些网站还提供 TeX 资料的电子信箱服务。下面是三个主要的 CTAN 网址：

`tug.ctan.org`，`http://www.ctan.org`（美国）

`ftp.dante.de`，`http://www.dante.de`（德国）

`ftp.tex.ac.uk`，`http://www.tex.ac.uk`（英国）

以下是几个 CTAN 镜像网址：

ctan.unsw.edu.au/tex-archive/（澳大利亚）
ctan.cms.math.ca/tex-archive/（加拿大）
ftp.funet.fi/pub/TeX/CTAN/（芬兰）
ftp.jussieu.fr/pub/TeX/CTAN/（法国）
ftp.nus.edu.sg/pub/docs/TeX/（新加坡）
ftp.kreonet.re.kr/pub/CTAN（韩国）
ftp.comp.hkbu.edu.hk/pub/TeX/CTAN/（中国香港）
ftp.riken.go.jp/pub/tex-archive/（日本）
dongpo.math.ncu.edu.tw/tex-archive/（中国台湾地区）
ftp.ctex.org/mirrors/CTAN/（中国）

第 2 章　源文件、命令及环境

大多数 TeX 文件的正文都是由字母构成的。若干字母合在一起组成单词，单词构成语句，语句构成段落，段落又构成更大的单位，如节和章等。

构成语句的单词之间由空格符或回车符隔开。TeX 把空格符和回车符解释为一个单词的结束，并且忽略单词之间更多的空格，多个空格也只当成一个空格来处理。段落之间是由一条或多条空行来分开的，而且一条空行与多条空行是等价的。段落之间的距离并不能由空行的多少来决定。TeX 视一个段落为一长串单词，选择合适的词距把它们从左至右地排列起来，并按照版面的宽度在适当的地方断行使段落左右对齐。断行处的选择是自动的，与源文件中断行无关。行距与所选字体的大小有关。段落由首行的缩格或增大的行距来区分。行距和段落之间的距离微有不同。TeX 及 LaTeX 按照版面的高度来适当地选择行距和段落之间的距离。分页处的选择像断行处的选择一样也是自动的。

LaTeX 系统在编译源文件时总处于三种模式之中，即段落模式 (paragraph mode)、数学模式 (math mode) 和左右模式 (LR mode)。段落模式是 LaTeX 中最普通、最常用的模式。在此模式中，LaTeX 把用户输入的文字看成由单词和句子组成并由空格分开的字符串，按照排版的要求自动将此字符串断行和分页。LaTeX 经由特别的命令切换到数学模式。在数学模式中，LaTeX 把用户输入的字符都看成是数学公式的一部分并忽略用户输入的空格。当遇到数学公式的结束命令时，LaTeX 从数学模式又切换到段落模式。左右模式类似于段落模式。在此模式中，LaTeX 仍然把用户输入的文字看成由单词和句子组成并由空格分开的字符串，只是排版时并不换行，而是从左至右一直排下去。经由特别的命令如 `\mbox` 可以切换到左右模式，即 LaTeX 将按照左右模式来处理命令 `\mbox{some text}` 中的文字 some text。另外，当一段文字被嵌入到数学公式中时也会出现左右模式。理解和认识 LaTeX 模式的重要性在于：有些命令只适用于特定的模式，而有些命令在不同的模式中有不同的作用。

最简单的情形是正文只包含输入的文字。TeX 处理这样的正文是按照标准的版面宽度和高度适当地选择行距、段落间距、断行处以及分页处使得行行对齐、页页一致。但事实上正文中都会包含大量的命令来通知 TeX 按照不同的要求排版。于是就有必要区分文字和命令。

2.1　LaTeX 的命令

LaTeX 命令 (command) 是区分大小写的，并且只有下述两种情形：

1) 一个反斜杠 \ 加上由若干字母组成的命令名，命令名终止于一个空格、数字或

其他非字母符号；

2) 一个反斜杠 \ 加上一个特殊符号。

LaTeX 忽略命令之后的空白。如果想得到命令之后的空白，就得在命令后加上{}，然后加上一个空格或一个输入空白的命令，以免这些空白被命令忽略。例如：

```
Knuth divides the people working
with \TeX{} into \TeX{}nicians
and \TeX perts.
```

Knuth divides the people working with TEX into TEXnicians and TEXperts.

有些命令需要一个参数，参数的内容必须置于命令名后面一对花括号{ }之中。有些命令的参数还附带若干选项，每个选项都置于命令之后一对方括号[ ]中。如下面两例所示：

```
This is a command
\textit{that need a parameter.}
```

This is a command *that need a parameter.*

```
For example, the parameter of the
command \framebox[2cm]{framebox}
has an option.
```

For example, the parameter of the command framebox has an option.

在后面我们将详细解释命令 \textit 和 \framebox。

LaTeX 还提供了两条命令 \newcommand 和 \renewcommand 用来定义新命令和更改已有命令，其用法见 2.15.1 节。

2.2 LaTeX 的环境

在 LaTeX 中，最重要，最方便的概念也许就是环境 (environment) 了。我们将看到大多数复杂的排版问题都可以通过设置 LaTeX 的环境而得到解决。

有些环境被用来用不同的方式显示文本的一部分，如 quote 环境和 verse 环境。center 环境可以用来使一段文字居中，而 flushleft 和 flushright 环境则可以使一小段文字分别左边对齐和右边对齐。另外还有 itemise、enumerate、tabular、figure 等许多其他环境，使我们能方便地排版列表、表格和图形。这都展现了 LaTeX 功能之强大。

环境的语法是：

\begin{*name*} *some text* \end{*name*}

其中 *name* 是环境的名称， \begin{*name*} 和 \end{*name*} 分别表示这个环境的开始与结束。因此，环境中的一切命令都是局部的，它们对环境外的文本不起作用。LaTeX

只将环境中的内容根据环境的定义进行排版。

环境可以相互嵌套，但内部的环境必须完全包含在外部环境中，如

```
\begin{aaaa}
外层环境中的内容
\begin{bbbb}
内层环境中的内容
\end{bbbb}
外层环境中的内容
\end{aaaa}
```

LaTeX 也提供了两条命令 \newenvironment 和 \renewenvironment 用来定义新环境和更改已有环境，见 2.15.2 节。

2.3 保 留 字 符

有 11 个字符是 TeX 和 LaTeX 的保留字符，它们要么有特殊的意义，要么不能适合所有字体。直接在源文件中输入这些字符，也许不能得到我们所期望的结果。这些字符是：

```
$ & % # _ { } ~ ^ \ |
```

它们中的前面 7 个可以通过在前面加上反斜杠 \ 来输出，如

```
\$ \& \% \# \_ \{ \}
```

$ & % # _ { }

另外，字符 ~ 和 ^ 可以分别用命令 \textasciitilde 和 \textasciicircum 或者分别用 \~{} 和 \^{} 来输出。字符 | 可以用 \textbar 或者 $|$ 来得到。反斜杠 \ 被称为命令前导符，它不能通过在它前面再加一个反斜杠来得到，而必须用其他方式，例如可以用 $\baskslash$ 或者用 \textbackslash 来输出反斜杠。反斜杠 \ 的意义我们已经清楚了，其他十个保留字符的意义在后文中将详细说明，现简要叙述如下：

- $ 进入和退出数学模式的符号；
- & 在表格环境中用来分隔一行中的各列；
- % 在源文件中用来注解前面的文本，一行中 % 后面的任何内容将被系统忽略；
- # #1,···,#9 用来定义命令的参数个数；
- _ 数学公式中用来定义下标；
- { 表示一个分组的开始；
- } 表示一个分组的结束；

~ 在西文中表示一个不可断行的空格；
^ 数学公式中用来定义上标；
| 常用于抄录环境和数学公式中。

2.4 源文件的结构

LaTeX 的源文件是一个纯文本文件，它必须包含导言部分 (preamble) 和主体部分 (body)。

导言部分通常是一些命令的集合，它们定义了文稿的整体结构，比如文稿的格式、版面的高度和宽度、行距和缩格、页面以及页码的形式、页眉、页脚以及旁注的形式等，也罗列了要用到的其他一些宏包。导言部分至少要包含 \documentclass 命令，它定义了整个文稿的类型是书本还是文章或者是其他形式的文本。这条命令通常也是整个文件的第一条命令。

如果导言部分除了 \documentclass 不包含其他的命令，那么 LaTeX 系统会选择内部设置的默认值来定义行宽、边距、段落间距、版面的高度和宽度等。这些设置一般只适合美国和西欧用户的习惯，其他国家和地区的用户也许要改变这些设置才合适。

导言部分的命令都具有整体效能，也就是说对于通篇文稿都有作用。

导言部分以命令 \begin{document} 为结束符，其后的部分就是正文。主体部分也包含一些命令，与导言部分中的命令不同的是它们只具有局部效能。主体部分以命令 \end{document} 为结束符，这也是整个文件的结束。大体上，LaTeX 源文件的基本结构如下：

```
\documentclass[options]{class}
此处罗列整体性的命令，对文稿进行整体设置。
\begin{document}
此处是正文和一些局部命令。
\end{document}
```

其中 *class* 和 *options* 的意义见 2.5 节。

2.5 源文件的类型

当 LaTeX 处理源文件时要知道的第一条信息就是该源文件是哪一类型的文件。这条信息就是通过命令 \documentclass 来传送的。此命令的语法是

\documentclass[*options*]{*class*}

其中 *class* 表示文件的类型。经常用到的文件类型有以下几种。

article 文章类型，通常用于排版科技文章、短篇报告等。

report 报告类型，通常用于排版有若干章的长篇报告、博士论文等。

book 书本类型，用于排版书籍。

letter 用于排版标准的书信。

slides 幻灯片类型，通常打印在透明胶片上，演讲时使用。这个类型使用大号 (big) sans serif 字体。

LaTeX2e 还提供了文件的其他一些类型，如 proc 类型和 ltxdoc 类型。命令中的 *options* 是文件类型的参数选项。两个选项之间必须用逗号“,”隔开。

例如，命令

```
\documentcalss[11pt,twoside,a4paper]{article}
```

提示 LaTeX 系统源文件的类型是 article，并要求按照 11pt 的字体大小用双页格式排印在 A4 纸张上。

下面列出了一些常用的文件类型的参数选项。

10pt, 11pt, 12pt 设置整篇文档的字体大小，默认设置是 10pt。

a4paper, letterpaper, ··· 定义纸张尺寸，默认设置是 letterpaper。另外还可选用 a5paper，b5paper，executivepaper 和 legalpaper。

fleqn 设置数学公式按照一定的缩格左对齐而不是居中。

leqno 把数学公式的标号放在左边而不是右边。

titlepage, notitlepage 决定标题之后的部分是否另起一页。article 类型的默认设置是标题后不另起一页，但 report 类型和 book 类型则另起新页。

twocolumn 设置 LaTeX 页面格式按双栏排版。

twoside, oneside 决定文稿双面排印还是单面排印。article 类型和 report 类型的默认设置是单面排印，而 book 类型的默认设置是双面排印的。注意这个参数只与文本的格式有关，而不是提示打印机是否按双面打印。

openright, openany 设置新的一章是从右页开始还是左右页都可以。这项设置对 article 类型无效，这是因为 article 类型没有定义章。report 类型的默认设置是 openany，而 book 类型的默认设置是 openright。

2.6 LaTeX 宏包及宏包套件

在编写 LaTeX 源文件时，基本的 LaTeX 命令可能不能解决用户遇到的问题，例如设置彩色文字、在文件中插入图片等问题。这时就需要一些扩展了 LaTeX 的某些功能

的文件，这种文件被称为 LaTeX 的宏包 (package)。我们可以用命令

\usepackage[*options*]{*package*}

来调用宏包，其中 *package* 是要调用的宏包的名称， *options* 是参数选项，它指出要调用的宏包的特别功能。下面是标准的 LaTeX2e 中包含的宏包。

alltt	这个宏包提供了 alltt 环境，环境中的字符除了 \, { 和 } 这三个字符具有通常的意义外，其他字符都按照打字机的字体排印。
doc	这是 LaTeX 的基本宏包，可用来排版 LaTeX 程序的说明文件。
exscale	这个宏包提供了按比例伸缩的数学扩展字体。
fontenc	指出使用哪种 LaTeX 字体编码。
graphpap	这个宏包定义了可在 picture 环境中使用的命令\graphpaper。
ifthen	提供了选择语句：“if ··· then do ··· otherwise do ···”。
inputenc	指出使用哪种 LaTeX 输入编码。
latexsym	LaTeX2.09 中的某些字符在 LaTeX2e 中不再自动装载，要使用这些字符可调用这个宏包。
makeidx	此宏包提供了排版索引的命令。
newlfont	用新字体选择方案模拟 LaTeX2.09 中的字体命令。
oldlfont	用于模拟 LaTeX2.09 中的字体命令。
showidx	它使源文件中的每个 \index 命令在输出页面上排印出来。
syntonly	用于编译源文件但不排版生成 dvi 文件，常用于检查源文件中的错误。
tracefnt	用于控制装载 LaTeX 字体信息的多少。

另外还有大量的宏包独立地存放于不同的地方。 Michel Goossens 等人编写的专著 The LaTeX Companion[7] 中介绍了大约 150 多个宏包的使用方法，以及如何编写宏包的有关知识。本书在各章节中也介绍了一些常用的宏包。

宏包套件就是由一系列宏包组成的套件。LaTeX2e 系统中附带了两个宏包套件：一个是工具宏包套件 (tools bundle)；另一个是图形宏包套件 (graphics bundle)。这两个宏包套件分别包含了许多有用的宏包，它们是标准 LaTeX 系统中宏包的改进或补充。有些系统（如 MiKTeX ）已把这两个宏包套件安装到 LaTeX2e 中，但有些系统（如 EMTeX）并未把它们安装到 LaTeX2e 中，需要用户自己安装。

2.7 长文件的编写

在编写有许多章节的长篇书本或文章时，一个比较好的方法是分别编写各章节并把它们保存在不同的文件中，然后在 LaTeX 源文件中调用这些文件。不需要将这些文件

分别编译，LaTeX 会将它们放入源文件中一并处理。LaTeX 中有两条命令帮助我们这么做。第一条命令是

\include{*filename*}

其中 *filename* 是一个不含扩展符的文件名。在 LaTeX 源文件的主体部分使用这条命令可以把名为 *filename.tex* 的文件中的内容插入到主体部分。要注意的是 LaTeX 编译这个文件的内容时会另起新页。第二条命令是

\includeonly{*filename1*,*filename2*,···}

这条命令必须用在源文件的导言部分。如果导言部分使用了这条命令，那么只有在这条命令中列出的文件的内容才有可能在使用 \include 命令时被插入到正文。注意，这条命令中的文件名与逗号之间不能有空格。命令 \include 总是开始一个新页，这有助于我们使用 \includeonly 这条命令，因为即使在源文件中，插入文件的地方遗漏了分页命令也无关紧要。不过，我们有时不希望在插入文件处另起新页。这时可以用命令

\input{*filename*}

这条命令直接把名为 *filename* 的文件的内容插入到源文件中而不附加任何条件。其中的文件可以不含扩展符，也可以含有扩展符。如果不含扩展符，那么系统会自动将名为 *filename.tex* 的文件插入到源文件中。在源文件的任何地方都可以使用这条命令。

为了使 LaTeX 系统快速地检查全文中的命令是否正确以及使用方法是否适当，我们可以在导言部分调用 snytonly 宏包，并在其后使用命令

\syntaxonly

若使用了这条命令，则在编译源文件时不会产生 dvi 文件，系统只是快速地检查了一遍全文并给出出错信息。当我们要得到 dvi 文件时，只需在上面这条命令同一行的前面加上一个百分号 %，然后再编译一遍就可以了。

2.8 文本的注释

在编辑 LaTeX 源文件时，为了增加源文件内容的可读性，我们常常要对文本进行注释 (comment)，而又不希望注释的内容被排印出来，这时我们可以使用字符 %。LaTeX 编译源文件时忽略字符 % 及其所在行后面的任何内容。这个字符的另外一个作用是分隔一个很长的输入行，这个行不允许有空格和断行。例如：

```
This is an % stupid
% Better: instructive <----
example: Supercal%
              ifragilist%
       icexpialidocious.
```

This is an example: Supercalifragilisticexpialidocious.

注意，% 只对一行中的字符起作用。如果我们有一大段的内容不想排印出来但又不舍得将这段内容删掉，以备将来可能要用到，这时当然可以在这个段落中每一行的开头加上 %，但如果这个段落很长，这么做就很费事了。利用 LaTeX2e 工具宏包套件中的宏包 verbatim 可以轻松办到。 verbatim 宏包提供了一个 comment 环境，它将此环境中的所有内容都注释掉。例如：

```
\begin{comment} With the comment
environment \end{comment}
it is possible to ignore passages
in the document during the
formatting process.
```

it is possible to ignore passages in the document during the formatting process.

注意，在 \end{comment} 所在行的后面不要包含其他用于排版的文字，这是因为 \end{comment} 就像百分号 % 一样不会把同一行中在它后面的文字排印出来。

2.9 显示源文本

排版中有时需要将一段文本不经过编译原封不动地显示出来。在 LaTeX 中的两个抄录环境：verbatim 环境和 verbatim* 环境

\begin{verbatim} 源文本 \end{verbatim}
\begin{verbatim*} 源文本 \end{verbatim*}

就能做到这一点。 LaTeX 将这两个环境中的源文本以它们原有的形式在一新行用打字机字体 (typewriter type face) 排印出来。环境中的源文本可以包含命令、特殊字符以及空格等所有字符，并且允许断行，这些内容在正常的文本中是要经过格式化编译才能排版的。这两个环境的不同之处在于带星号的环境将空格以符号 ␣ 的形式打印出来，以利于弄清源文本中包含的空格。例如：

```
\begin{verbatim*}
LaTeX is a very powerful
formatting program!
\end{verbatim*}
```

```
LaTeX␣is␣a␣very␣powerful
formatting␣program!
```

我们也可以用下面的两条命令

```
\verb符号 源文本 符号
\verb*符号 源文本 符号
```

将较短的源文本在一行中原封不动地显示出来，其中的符号是任何不出现于源文本中的符号，当然它不能是星号 *，而且源文本前后的这一对符号必须是一样的。另外，应当注意源文本中不能断行。带星号的命令 \verb* 同带星号的 verbatim* 环境一样也将空格以符号 ␣ 的形式打印出来。

上面两种 verbatim 环境和两种 \verb 命令都不能用于其他命令中，否则可能产生出错信息。

2.10 长 度

在 LaTeX 中经常要用到长度的概念。最简单的长度就是一个十进制数（可正可负）加上一个长度单位。下面列出了 LaTeX 中常用的一些长度单位。

mm 毫米 1毫米 ≈ 1/25 英寸

cm 厘米 1厘米 = 10 毫米

in 英寸 1英寸 = 25.4 毫米

pt 点 1pt = 1/72.27 英寸 ≈ 1/3 毫米

bp 大点 1bp = 1/72 英寸

pc pico 1pc = 12pt

em 相当于当前字体的大写字母“M”的宽度

ex 相当于当前字体的小写字母“x”的高度

用来定义长度的参数称为长度参数，它必须是带反斜杠的命令形式。在 LaTeX 中有两种类型的长度参数，即刚性长度参数和弹性长度参数。刚性长度参数具有固定的值而弹性长度参数则没有固定值，它的值随着情况的不同而有所改变。所有 LaTeX 的标准长度参数都是刚性的。命令

```
\fill
```

提供了一个弹性长度，它的原始值是零但可以根据不同情况自动伸长到任意一个正值。另外，命令

```
\stretch{n}
```

也提供了一个弹性长度，其中 n 是一个十进制数，表示伸缩因子。\stretch{n} 定义的长度是 \fill 的 n 倍，因此 \fill 等价于 \stretch{1} 。

另外，形如 20pt plus 3pt minus 2pt 的长度也是一种有一定伸缩性的弹性长度，比如这个长度就表示 20pt，但在必要时可伸长到 23pt，或缩短到 18pt。

为了准确地得到排版结果，标准 LaTeX 本身已经提供了大量的长度参数，例如 \textheight, \textwidth 等，但有时仍需要定义一些新的长度参数。下面的命令

\newlength{*cmd*}

定义 *cmd* 是一个新的长度参数。如果命令 *cmd* 已经存在，那么编译时就会出错。长度参数一旦定义好了，它的值就被自动设置为 0pt 。所有刚性长度参数以及新定义的长度参数都可以重新赋值。下面这条命令

\setlength{*cmd*}{长度}

就是用来对长度参数重新赋值的，它把 *cmd* 的值设置为长度。如果需要将一个长度加到长度参数原有的值上，可以用命令

\addtolength{*cmd*}{长度}

它把长度加到了 *cmd* 的原有的值上。有时我们要将几个字的自然宽度取出来用在别处，这时可以用命令

\settowidth{*cmd*}{文本}

此命令将文本这两个字组成的整体的自然宽度取出来赋给 *cmd* 。下面的两条命令

\settoheight{*cmd*}{文本}　和　\settodepth{*cmd*}{文本}

与 \settowidth 有类似的作用，只是取出来的分别是文本的高度和深度。

2.11 空　　白

在编辑 LaTeX 或 TeX 源文件时经常要使用空白，如字距、行距、段落间距等。通常 LaTeX 是自动设置这些空白的，若要改变空白的大小，就必须在源文件中添加适当的命令。

2.11.1 水平空白

在 LaTeX 文件中可以用下面的命令来改变水平方向的空白距离：

\hspace{*length*}

其中 *length* 是空白距离的长度。如果要在章节的开头或行首增加空白距离可以用带星号的命令 \hspace* 来代替 \hspace。例如：

```
\hspace*{1cm}This \hspace{1.5cm}
is a space of 1.5 cm.
```

This is a space of 1.5 cm.

下面这条命令

\hspace{\stretch{n}}

可以产生一个可伸缩的弹性水平空白，它将对行中的空白进行拉伸直至整行都被填满。命令中的 n 是一个十进制数，它表示的是伸缩因子。如果在一行中写了两条形如 \hspace{\stretch{n}} 的命令，则行中的空白将会根据伸缩因子进行分配。如下面例子所示：

```
This is a \hspace{\stretch{1}}
example.
```

This is a example.

此例中只有一个 \hspace{\stretch{1}}，系统会将命令所在处的空白拉伸，命令中的因子是 1 还是其他的数无关紧要，但不可省去。下面例子中的因子是起作用的，它表示后面的空白是前面的空白的 3 倍。

```
This is \hspace{\stretch{1}} a
\hspace{\stretch{3}} example.
```

This is a example.

虽然也可以使用负的因子来调整空白的分配，但还是尽量不使用，这是因为它有可能改变文字的顺序。

2.11.2 垂直空白

类似于段落、小节、节和章之间的空白距离称为垂直空白。LaTeX 是自动调整这些垂直空白距离的。但我们可以根据需要使用下述命令

\vspace{*length*}

来增加或减少段落之间的垂直空白。一般来说这条命令应置于两条空行之间。如果要在页面的顶部或者底部增加垂直空白，应该用带星号的命令 \vspace* 来代替 \vspace。

利用命令 \stretch 并结合命令 \pagebreak 我们可以将一段文字排在页面的顶部、底部或者中间。

在同一个段落的两行或者表格中的两行之间可以使用命令 \\[*length*] 来改变两行间的距离，其中的 *length* 是一个长度。

2.12 特 殊 字 符

2.12.1 引号

写 LaTeX 源文件的时候要注意不应该使用 " 来作为引号。在印刷界有特殊的引用语开始标记和结束标记。 LaTeX 使用两个‘来作为左边双引号，用两个，来作为右边双引号。例如：

```
``Please press the `x' key.''
```

“Please press the ‘x’ key.”

命令 \textquoteleft 和命令 \textquoteright 也分别排印出左边引号“‘”和右边引号“’”。

2.12.2 破折号与连字符

LaTeX 中有三种破折号 (dash)，我们可以通过输入不同个数的连字符来输出。第一个破折号就是连字符 (hyphen)，用于复合词中或者用于分割单词；第二个称为数字连接符 (en-dash)，用于表示一个数字区域，它可以用两个连字符或者用命令 \textendash 来输出；第三个也就是通常的破折号 (em-dash)，用于复合语句，它可以用三个连字符或者用命令 \textemdash 来输出。还有一个是数学中的减号，可用 $-$ 来输出。例如：

```
daughter-in-law, X-rated\\
pages 13--67\\
yes---or no? \\
$0$, $1$ and $-1$
```

daughter-in-law, X-rated
pages 13–67
yes—or no?
0, 1 and −1

2.12.3 省略号

在打字的时候，逗号和句号所占的空间与其他字母所占的空间是一样的，但是直接排印出来之后这些符号所占的空间非常小且与后面的字母很靠近。因此，我们不能用连续的三个句号来排版省略号，否则三个点之间的间距就不正确。在 LaTeX 中我们用下面的命令

```
\ldots
```

来输出省略号。这条命令输出的省略号是排在基线上的，如果要得到排中间的省略号，可以用排版数学公式的命令 \cdots。例如：

```
Not like this ... but like this:\\
New York, Tokyo, Budapest, \ldots \\
like this $\cdots$
```

Not like this ... but like this:
New York, Tokyo, Budapest, ...
like this ···

2.12.4 连体字母

某些字母连在一起写的时候，排版的结果并不像通常一样一个接着一个地排，而是用一些将它们结合在一起的特殊符号。例如，我们输入“`ff fi fl ffi`”，排出的结果却是“ff fi fl ffi”，而不是“ff fi fl ffi”。我们看到，在前一种情形中，两个或者三个字母连在了一起。这种几个连在一起的字母称为连体字母 (ligature)。如果一个词中出现过多的连体字母就显得不太好看，如 shelfful，此时可以在字母之间插入命令 `\mbox{}` 或者 `\/`，也可以插入命令 `\textcompwordmark`，这样就可以使字母之间稍微留些空隙。

2.12.5 特殊外文字符及重音

在一些欧洲国家的语言中存在与英语不同的字母，它们通常不能从键盘直接输入。LaTeX 提供了特殊的命令来排印这些字母。它们是

```
œ={\oe}  Œ={\OE}  æ={\ae}  Æ={\AE}  å={\aa}  Å={\AA}  ß={\ss}
ø={\o}   Ø={\O}   ł={\l}   Ł={\L}   ¡={!‘}   ¿={?‘}
```

另外，许多外文字母的上面或者下面常需要加重音符号。在标准 LaTeX 中有各种对字母加重音的命令。下面列出了用于字母 o 的所有重音符号及相应的 LaTeX 命令，这些命令用在其他字母也是有效的。

```
ò=\‘o   ó=\’o   ô=\^o   õ=\~o   ō=\=o   ȯ=\.o   ö=\"o
ǒ=\u{o} ǒ=\v{o} ő=\H{o} o̧=\c{o} ọ=\d{o} o̲=\b{o} o͡o=\t{oo}
```

要注意在字母 i 和 j 上加重音，必须把这两个字母上的点先去掉，这可用 `\i` 和 `\j` 来完成。例如：

```
H\^otel, na\"\i ve, \’el\‘eve,\\
sm\o rrebr\o d, !‘Se\~norita!,\\
Sch\"onbrunner Schlo\ss{} Stra\ss e
```

Hôtel, naïve, élève,
smørrebrød, ¡Señorita!,
Schönbrunner Schloß Straße

此外，命令 `\r{字符}` 可以在任何字符上面加一个小圆圈形式的重音。如命令 `\r{o}` 可排印出 o̊。

2.12.6 美元英镑等符号

还有一些特殊的字符键盘上不存在，而要用特殊的命令才能排印出来。例如：

§ = \S, † = \dag, ‡ = \ddag, ¶ = \P, © = \copyright, £ = \pounds

在 LaTeX2e 中命令 \textcircled{字符} 可以排印出像 © 一样带圈的字符，例如 ⓐ 可以用 \textcircled{a} 来排出。

2.12.7 可见空格符

在一些计算机列表和说明文件中常需要将空格字符显示出来以利于阅读和理解。在 LaTeX2e 中，命令 \textvisiblespace 可排印出符号 ␣，它代表一个可见空格。例如

```
hello\textvisiblespace{}world
```

排印出 hello␣world。可见空格符也可以用命令 \verb*| | 来显示。

2.12.8 bbding 符号

有些符号如 ✂、✪、❀ 等，以前在 LaTeX 中的使用比较麻烦，需要用到 PostScript 字体（如 pifont 等），而且有的 dvi 预览程序（如 MiKTeX 中的 Yap）还不能预览。Karel Horak 编写的 bbding 宏包提供了一种简单的方法来排印这种 bbding 符号。当系统安装了这个宏包之后，只需在源文件的导言部分加入 \usepackage{bbding} 这条命令我们就可以在正文中排印各式各样的 bbding 符号了。表 2.1 至表 2.8 列出了这个宏包提供的所有符号及相应的命令。

表 2.1 花朵类符号

符号及其命令	符号及其命令
❀ \FiveFlowerOpen	✿ \FiveFlowerPetal
✲ \SixFlowerOpenCenter	❀ \SixFlowerRemovedOpenPetal
❋ \SixFlowerAlternate	❋ \SixFlowerAltPetal
❀ \SixFlowerPetalDotted	✲ \SixFlowerPetalRemoved
✳ \EightFlowerPetalRemoved	✿ \EightFlowerPetal
❀ \FourClowerOpen	✤ \FourClowerSolid
✳ \Sparkle	✳ \SparkleBold
❄ \SnowflakeChevron	❄ \SnowflakeChevronBold
❄ \Snowflake	

表 2.2 星形类符号

符号及其命令	符号及其命令	符号及其命令
✡ \DavidStar	✧ \FourStarOpen	✳ \AsteriskThinCenterOpen
✦ \FourStar	✤ \JackStarBold	✸ \DavidStarSolid
★ \FiveStar	☆ \FiveStarLines	☆ \FiveStarCenterOpen
✶ \SixStar	✸ \EightStarBold	✭ \FiveStarOpenDotted
✷ \EightStar	☆ \FiveStarShadow	✭ \FiveStarConvex
✻ \Asterisk	★ \FiveStarOutline	✭ \FiveStarOutlineHeavy
☆ \FiveStarOpen	✷ \EightStarTaper	✷ \EightStarConvex
✹ \TwelweStar	✺ \SixteenStarLight	✪ \FiveStarOpenCircled
✱ \AsteriskBold	✳ \AsteriskThin	✲ \AsteriskCenterOpen
✣ \JackStar	✵ \EightAsterisk	✿ \AsteriskRoundedEnds
✢ \FourAsterisk		

表 2.3 剪刀类符号

符号及其命令	符号及其命令
✂ \ScissorRight	✁ \ScissorRightBrokenBottom
✁ \ScissorLeft	✃ \ScissorLeftBrokenBottom
✄ \ScissorHollowRight	✁ \ScissorRightBrokenTop
✄ \ScissorHollowLeft	✃ \ScissorLeftBrokenTop

表 2.4 十字架类符号

符号及其命令	符号及其命令	符号及其命令
╳ \XSolid	✖ \XSolidBold	✗ \XSolidBrush
✚ \Plus	✚ \PlusOutline	✙ \PlusCenterOpen
✝ \PlusThinCenterOpen	† \Cross	✞ \CrossOpenShadow
✝ \CrossOutline	✝ \CrossBoldOutline	✣ \CrossClowerTips
✠ \CrossMaltese		

表 2.5 铅笔类符号

符号及其命令	符号及其命令	符号及其命令
\PencilRight	\PencilRightUp	\PencilRightDown
\PencilLeft	\PencilLeftUp	\PencilLeftDown
\NibRight	\NibSolidRight	\NibSolidLeft
\NibLeft		

表 2.6 杂项符号

符号及其命令	符号及其命令	符号及其命令
\Phone	\PhoneHandset	\ArrowBoldRightStrobe
\Plane	\Checkmark	\ArrowBoldRightShort
\Envelope	\CheckmarkBold	\SunshineOpenCircled
\Tape	\ArrowBoldUpRight	\ArrowBoldRightCircled
\Peace	\ArrowBoldDownRight	

表 2.7 几何图形类符号

符号及其命令	符号及其命令
\CircleSolid	\CircleShadow
\HalfCircleRight	\HalfCircleLeft
\Ellipse	\EllipseSolid
\EllipseShadow	\Square
\SquareSolid	\SquareShadowBottomRight
\SquareShadowTopRight	\SquareShadowTopLeft
\SquareCastShadowBottomRight	\SquareCastShadowTopRight
\SquareCastShadowTopLeft	\TriangleUp
\TriangleDown	\DiamondSolid
\OrnamentDiamondSolid	\RectangleThin
\Rectangle	\RectangleBold

表 2.8 手形类符号

符号及其命令	符号及其命令	符号及其命令
☞ \HandRight	☟ \HandRightUp	☛ \HandCuffRight
☟ \HandCuffRightUp	☜ \HandLeft	☚ \HandLeftUp
☚ \HandCuffLeft	☝ \HandCuffLeftUp	✍ \HandPencilLeft

2.13 计 数 器

在 LaTeX 中有许多地方都要用到编号，例如页码、章节号、公式编号、图形和表格编号等。大多数这种编号都是通过称为计数器的变量来自动产生的。不同性质的编号由不同名字的计数器生成。计数器的名字通常与命令或环境的名字相同只是不带反斜杠 \。计数器的值只能是整数。几种不同计数器可以合成各种复杂的数，如 2.3.1 节等。在使用 LaTeX 排版文稿的过程中还经常需要定义新的计数器来对某些新内容进行编号。下面列出了标准 LaTeX 文件类型中的所有计数器名：

```
part            paragraph      figure      enumi
chapter         subparagraph   table       enumii
section         page           footnote    enumiii
subsection      equation       mpfootnote  enumiv
subsubsection
```

另外，由命令 \newtheorem 定义的每一个定理表述环境都会有一个相应的计数器来记录定理的序号。

下面这条命令

\newcounter{*newctr*}[*oldctr*]

可以用来定义一个名为 *newctr* 的新计数器，且它的初始值为零。命令中的 *oldctr* 是可选参数，它是指定的已经存在的旧计数器名。如果命令中含有这类可选参数的话，那么每当计数器 *oldctr* 的值增加 1 时，计数器 *newctr* 的值就被重新设置为零。此时我们称 *newctr* 是 *oldctr* 的子计数器。下面两条命令

\setcounter{*ctrname*}{*val*} 和 \addtocounter{*ctrname*}{*val*}

用来改变计数器 *ctrname* 的值，其中第一条命令将计数器的值设置为 *val* 所表示的数，而第二条命令将 *val* 所表示的数加到计数器原有的值上， *val* 是整数或者是命令

`\value{`*ctr*`}`

它表示计数器 *ctr* 的值。命令

`\stepcounter{`*ctrname*`}`

将计数器 *ctrname* 的值增加 1，因而它的所有子计数器的值都被重新设置成零。命令

`\refstepcounter{`*ctrname*`}`

同 `\stepcounter` 一样也将计数器 *ctrname* 的值增加 1，但它同时将 *ctrname* 作为 `\label`（此命令在交叉引用中用于打标签）的当前计数器，因此当用命令 `\ref` 来引用这个标签时，实际上是引用 *ctrname* 的值。命令

`\thectr`

用来打印计数器 *ctr* 的值。例如：

```
\newcounter{ctrA}\newcounter{ctra}[ctrA]
\setcounter{ctrA}{3}\setcounter{ctra}{1}
\refstepcounter{ctrA}
\stepcounter{ctrA}\label{A}
ctra=\thectra\quad
ctrA=\thectrA\quad ref=\ref{A}
```

ctra=0 ctrA=5 ref=4

计数器可以有不同的表现形式，如阿拉伯数字、罗马数字、英文字母等。下面列出了用来表现计数器形式的命令：

`\arabic{`*ctr*`}`	将计数器 *ctr* 表示为阿拉伯数字；
`\roman{`*ctr*`}`	将计数器 *ctr* 表示为小写罗马数字；
`\Roman{`*ctr*`}`	将计数器 *ctr* 表示为大写罗马数字；
`\alph{`*ctr*`}`	将计数器 *ctr* 表示为小写英文字母，其值不能超过 26；
`\Alph{`*ctr*`}`	将计数器 *ctr* 表示为大写英文字母，其值不能超过 26；
`\fnsymbol{`*ctr*`}`	将计数器 *ctr* 表示为脚注符号，如 *，† 等，此命令只能用于数学模式中。

2.14 盒 子

盒子 (box) 是 TeX 系统用于排版的基本单位，它是一个矩形区域，具有三个度量，即高度、宽度和深度。图 2.1 展示了一个带边框的盒子及其参照点和基线。对 TeX 系统来说，每种字体的每一个字符都是一个盒子，这是一种最简单的盒子。字体设计者已将这种盒子的高度、宽度和深度确定了。TeX 系统用优化的方法来选择参照点和基线将这些字符盒子排成一行。若干个字符盒子就组成了以词为单位的盒子如 Word。这些以词为单位的盒子之间插入弹性距离后再排成一行就组成了以行为单位的盒子，称为行盒子。若干行盒子沿垂直方向排列形成段落盒子，段落盒子又被置于主体盒子之中，主体盒子连同页眉和页脚盒子一起最后构成了页面盒子。

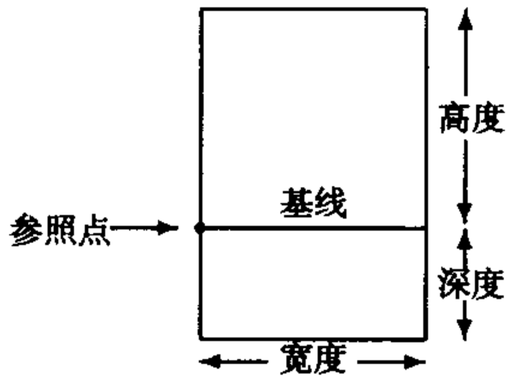


图 2.1 盒子示例

每一个盒子都是一个整体。TeX 系统排版时视其如单个字符一样不再将其分裂开来，尽管它可能是由一些更小的盒子组成。不过根据排版的需要也可以将整个盒子上下左右移动。

在 LaTeX 系统中，用户可以选用三种类型的盒子，它们是左右盒子 (LR box) 、段落盒子 (paragraph box) 和标尺盒子 (rule box)。左右盒子中的内容处于左右模式中，它们是由一些从左至右水平排列的材料组成的；段落盒子是由若干个行盒子沿垂直方向排列而成的一个整体；标尺盒子则是一个用黑色填充的矩形区域，通常用于画垂直或水平直线。

2.14.1 左右盒子

在 LaTeX 中，下面两条命令

```
\mbox{文本}  和  \fbox{文本}
```

都能创建一个内容为文本的左右盒子。其中第一条命令 \mbox 所创建的盒子的宽度恰好是花括号内文本的宽度。第二条命令 \fbox 则创建一个带有边框的左右盒子，输出形式是 文本 ，其宽度和总高度要比文本自身所占有的宽度和总高度更多一些。如果要创建一个具有预先指定宽度的盒子可以用下面两条命令

```
\makebox[宽度][位置]{文本}  和  \framebox[宽度][位置]{文本}
```

其中第一条命令创建一个不带边框的盒子，而第二条命令与 \fbox 一样创建带边框的盒子。

命令中的 宽度 是可选参数，表示所创建的盒子的宽度，它是一个长度，如 2cm、0pt 等，必须带有单位。另外，也可以取相对宽度。所谓相对宽度就是当使用 \mbox 时出现的自然宽度、高度、深度或总高度的一个倍数。这些自然宽度、高度、深度、总高度是：

\width	使用 \mbox 时的文本原始宽度；
\height	使用 \mbox 时的文本原始高度，即基线到顶部的距离；
\depth	使用 \mbox 时的文本原始深度，即基线到底部的距离；
\totalheight	就是 \height 加上 \depth。

例如，输入命令 \framebox[6\height]{hello}，将得到一个包含 hello 且宽度是自然高度 6 倍的盒子 hello 。注意,这些相对宽度命令只能用在盒子的宽度定义中，不能用于其他的地方，否则会产生错误信息。

上面命令中的 位置 也是可选参数，它规定盒子中的文本在盒子中是居右、居左还是居中排版。可以用下面几个字母来表示这个参数的值：

l	盒子中文本居左；
r	盒子中文本居右；
c	盒子中文本居中，这是默认选项；
s	盒子中文本从左向右伸展以占满整个盒子。

例如：

```
\framebox[6cm][r]{right}\\[1mm]
\framebox[6cm][s]{s p r e a d}
```

right

s p r e a d

如果盒子中文本的实际宽度要比所定义的盒子宽度更宽，那么盒子中文本视其位置的不同将向左、向右或在两个方向超出盒子的边界。这种情况通常不是我们所希望的，但也可以利用这一点来使两部分文本重叠，从而达到一种特殊的效果。对于一个盒子来说，如果我们把参照点所在的边称为起始边，另一条相对的边称为终止边，那么当位置参数为 l 时，盒子中文本的第一个字符是靠在起始边右侧的，而当位置参数为 r 时，盒子中文本的最后一个字符就紧靠在终止边左侧。盒子的宽度为零（如 0pt ）时，终止边实际上与起始边重合。由于盒子后面的文字总是紧接着终止边排版的，因此当盒子的终止边与起始边重合时，盒子后面的文字就要与盒子中的文字重叠。

零宽度的盒子常被用来在页面的某处做标记或者在页面的两边空白处做旁注，盒子中的内容不会影响盒子外面的排版。在下面的例子中，我们将三行文字居中排版，并在第一行的右上角和第三行左上角做了标记，这些标记没有影响文字的居中。注意，盒子的位置参数 [l] ([r]) 使盒子中内容放在右边（左边）：

```
\begin{center}
A centered sentence.%
\makebox[0pt][l]{$^{123}$}\\
Some more text in the middle.\\
\makebox[0pt][r]{$^{321}$}%
A centered sentence.
\end{center}
```

A centered sentence.[123]
Some more text in the middle.
[321]A centered sentence.

在下面这个例子中，我们利用一个零宽度的盒子将一个箭头放置在一个段落的左边空白处：

```
\makebox[0pt][r]{$\Longrightarrow$}Now
We use a box with zero width to put an
arrow on the left of the beginning.
```

⟹Now We use a box with zero width to put an arrow on the left of the beginning.

对一个负宽度的盒子来说，终止边在起始边左侧且到起始边的距离恰好是同样大小的正宽度的距离。也就是说，负宽度的盒子就是将原来正宽度的盒子以起始边为轴做了一个镜面反射而得到的。例如，输入命令 =\makebox[-0.3\width][r]{/}\，可使斜杠 / 向左移动一些与等号重合，从而产生一个不等号 ≠。输入命令

```
S\makebox[-0.6\width][r]{\textbar}
```

可得到符号 $。

应当注意的是左右盒子中不能包含 \verb 命令和 verbatim 环境。

对于由命令 \fbox 和 \framebox 产生的带边框的盒子我们可以用下面的命令

\fboxrule 和 \fboxsep

来改变方框线的粗细程度以及方框线到方框内文字边缘的距离。这两条命令都是长度参数命令，可以使用 \setlength 命令对它们重新赋值。例如：

```
\setlength{\fboxrule}{0.4pt}
\setlength{\fboxsep}{1mm}
```

前一条命令将方框线的粗细设置为 0.4pt，而后一条命令则将方框线到方框内文字边缘的距离设置为 1 毫米。

2.14.2 左右盒子的垂直位移

如果需要将一个左右盒子向上移动或者向下移动，那么我们可以用下面的命令

\raisebox{位移量}[高度][深度]{文本}

此命令创建一个包含文本的左右盒子。盒子的高度是高度，深度是深度，且基线被垂直移动了一个长度为位移量的距离。位移量为正距离表示向上移动，为负距离表示向下

移动。如果高度和深度这两个可选项被省略，则取文本原本所在盒子的相应值。位移量、高度和深度都是长度，可以取通常的长度如 2cm, 也可以取上一节中提到的 \width, \height 等相对长度。例如：

```
Baseline \raisebox{1ex}{high} and
\raisebox{-1ex}{low} and back again.
Some other text in second line.
```

Baseline $^{\text{high}}$ and $_{\text{low}}$ and back again. Some other text in second line.

为了不使前后行的文字相互重叠， LaTeX 在排版中是根据一行中所有盒子的高度和深度来确定前后行距的。因此，如果在一行中上移或下移一个盒子而又指明此盒子的高度和深度不大于这一行中其他盒子的高度和深度，那么被移动的盒子就有可能与前行或后行重叠。

2.14.3　左右盒子的储存与提取

在使用 LaTeX 排版文稿时，如果有一部分内容或文本（特别是图形）要多次被用到，那么就有必要用下面的方法

\newsavebox{\boxname}

来定义一个名为 \boxname 的储存盒子，注意储存盒子的名字必须是带反斜杠的标准命令形式。然后将多次被使用的文本通过命令

\sbox{\boxname}{文本}　或者　\savebox{\boxname}[宽度][位置]{文本}

保存在所定义的储存盒子中备用，这里宽度及位置与 \makebox 和 \framebox 中的宽度及位置有相同的意义。要特别注意此处被保存的内容中不能含有 \verb 命令和 verbatim 环境。最后，当要使用保存的内容时，只要调用命令

\usebox{\boxname}

就可以了。另外在 LaTeX2e 中还可以通过下面的 lrbox 环境

\begin{lrbox}{\boxname} 文本 \end{lrbox}

将文本保存在名为 \boxname 的储存盒子中。此时被保存的内容（即 文本）前后的空格将被省略。这种利用 lrbox 环境来保存将被多次使用的内容的方法实际上有更广泛的用处。因为在这里被保存的内容几乎可以是任何形式的文本，包括 \verb 命令和 verbatim 环境。这使得我们可以用 lrbox 环境来方便地定义一些新的特殊形式的环境。定义新环境的方法可参见 2.15.2 节的内容。

2.14.4 段落盒子及小页环境

下面的 \parbox 命令和 minipage 环境（即小页环境）

```
\parbox[位置]{宽度}{文本}
\begin{minipage}[位置]{宽度} 文本 \end{minipage}
```

都能创建一个同等效果的具有所给宽度的段落盒子，其中可选参数位置表示所创建的段落盒子相对于外部文字（即相对于命令或环境所在行的基线）的垂直摆放位置，它可以取如下几个值：

- t 表示段落盒子的顶部与外部文字的基线对齐；
- c 表示段落盒子的中间与外部文字的基线对齐，这是默认值；
- b 表示段落盒子的底部与外部文字的基线对齐。

下面是一个简单例子：

```
\HR{} \begin{minipage}[b]{15mm}
A A A A A A A A A A A A A A A
\end{minipage} \HR{}
\begin{minipage}[c]{12mm}
B B B B B B B B B B B B
\end{minipage} \HR{}
\begin{minipage}[t]{14mm}
C C C C C C C C C C C C C C
\end{minipage} \HR{}
```

```
A A A A
A A A A
A A A A  B B B B
_A A A   _B B B B_C C C C_
           B B B B  C C C C
                    C C C C
                    C C
```

其中盒子所在的基线用由 \HR 所定义的一条短横线显示出来。这里 \HR 的定义是

```
\newcommand{\HR}{\rule{3mm}{0.3pt}}
```

这个例子中我们将三个小页并排摆放，第一个小页的底部基线和第三个小页的顶部基线都与中间小页中部的基线对齐。

在 LaTeX 排版系统中，minipage 环境是一个非常有用的环境。事实上，它是一个缩小的页面，其中可以包含自己的脚注、段落、表格等，也可以在里面进行多栏排版。还可以在 minipage 环境中使用 \verb 命令和 verbatim 环境，但是里面不能包含浮动体和旁注。

如果要得到比较复杂的对齐方式，可能需要把不同的 minipage 环境以不同的方式组合起来使用。下面我们试图把左边两个字母块在顶部对齐，所形成的大块的底部又与第三个字母块的底部对齐：

```
\HR{} \begin{minipage}[b]{36mm}
\begin{minipage}[t]{15mm}
A A A A A A A A A A A A A A
\end{minipage} xx
\begin{minipage}[t]{12mm}
B B B B B B B B B
\end{minipage}
\end{minipage} \HR{}
\begin{minipage}[b]{15mm}
C C C C C C C \end{minipage} \HR{}
```

```
                                        C C C C C
_ A A A A xx B B B B   _ C C       _
  A A A A    B B B B
  A A A A    B
  A A A
```

但是按照这样来排版我们并没有得到设想的结果，原因是左边两个顶部对齐的字母块所形成的大块被 TeX 看成是 xx 所在行上的一个形状大些的字符，所以这个单独的行是字母块的顶行的同时也是大块的最后一行。要避免这种情况，我们需要在左边两个字母块的后面加入一个由 \par\vspace*{0pt} 定义的不可见的虚段落，使这个大块的最后一行移到底部。

```
\HR{} \begin{minipage}[b]{36mm}
\begin{minipage}[t]{15mm}
A A A A A A A A A A A A A A
\end{minipage} xx
\begin{minipage}[t]{12mm}
B B B B B B B B B
\end{minipage}\par\vspace*{0pt}
\end{minipage} \HR{}
\begin{minipage}[b]{15mm}
C C C C C C C \end{minipage} \HR
```

```
  A A A A xx B B B B
  A A A A    B B B B
  A A A A    B          C C C C C
_ A A A                 _ C C       _
```

2.14.5　有确定高度的段落盒子

在排版中有时需要一个有确定高度的段落盒子，为此 LaTeX2e 提供了有更多选项的 \parbox 命令

```
\parbox[位置][高度][内部位置]{宽度}{文本}
```

和 minipage 环境

```
\begin{minipage}[位置][高度][内部位置]{宽度}
文本
\end{minipage}
```

其中位置和宽度两个参数与上一小节相同。可选参数内部位置表示段落盒子中的文本在盒子中所处的垂直意义上的位置，它可以取如下的值：

- t 表示文本被排印在盒子的顶部，即文本首行与盒子顶部对齐；
- c 表示文本被排印在盒子的中间，即文本垂直中心与盒子垂直中心对齐；
- b 表示文本被排印在盒子的底部，即文本最后一行与盒子的底部对齐；
- s 表示文本从顶部向底部伸展开来占满整个盒子。

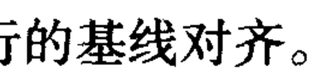
可选参数高度表示段落盒子的高度，它既可以是通常的长度（如 4cm, 5in 等），也可以同 \makebox 和 \framebox 这类命令中的宽度参数一样使用相对长度参数，如：\height、\width、\depth 和 \totalheight。高度的值如果设置的太小，那么文本有可能从顶部、底部或上下两边超出盒子的边界。

另外，要注意的是不能单独省略位置这个可选参数。如果内部位置被省略了，那么它的值就与位置的值相同。如果内部位置和位置都被省略了，那么所定义的高度也就不起作用。如果高度被省略，那么内部位置也会失去意义。因此也不能单独省略高度这个选项。

2.14.6 标尺盒子

标尺盒子就是用黑色填充的矩形区域，它可以由命令

```
\rule[垂直位移量]{宽度}{高度}
```

来创建，其中 宽度 和 高度 分别表示盒子的宽度和高度。垂直位移量 表示盒子向上或向下的位移量（正数表示向上移动而负数表示向下移动），此参数省略时盒子的底部与命令所在行的基线对齐。例如命令 \rule{8mm}{4mm} 的结果是  ，而命令 \rule[-1mm]{8mm}{4mm} 的结果是  ，这里的短横线表示基线。标尺盒子常被用于画横线和竖线。然而宽度为零或高度为零时将会产生不可见的标尺盒子，它们有特别的用处。宽度为零的标尺盒子又称为支柱(strut)，它可以用来增加其所在盒子的高度和深度。\vspace 就不能起到这种作用，因为它只能用来增加已有的垂直间距。例如命令 \fbox{text} 的结果为 text，但用命令 \fbox{\rule[-1.5mm]{0mm}{5mm}text} 可以得到 text，text 所在盒子向基线上下伸展了一定长度，这是因为盒子中包含了一个不可见的支柱将盒子的上下两边支撑起来。标尺盒子的这种作用常被用来局部调整表格中行与行之间的距离。另外高度为零的标尺盒子能够将它所在盒子的左右两边向外伸展一定的距离。但这个性质没有多大效用，因为我们可以使用命令 \hspace 来达到同样的效果。

2.14.7 盒子的嵌套

因为 TeX 将盒子看成是一个字符来对待，而盒子中又可以包含字符，所以盒子是可以嵌套使用的。

> 把一个段落盒子放在一个带边框的左右盒子之中就得到一个带边框的段落盒子，再把这个带边框的段落盒子放在一个带边框的左右盒子之中就得到一个带双层边框的段落盒子。

例如，上面方框里的内容就是用如下形式的命令

```
\fbox{\fbox{\parbox{12cm}{把一个……}}}
```

产生的。

把一个段落盒子放在 \raisebox 中就可以将此盒子整个向上移动或者向下移动。当我们把段落盒子放在 \sbox 或 \savebox 中时就可以将此盒子保存起来，方便以后再次使用。总之，对盒子进行各种各样的排列和嵌套就可以得到各式各样的排版效果。这种方法正是 TeX 排版系统的实质。

2.14.8 不同边框的盒子

fancybox 宏包定义了四种命令来排印具有不同边框的左右盒子，盒子中的文本到边框的距离与 \fbox 一样也是由长度参数 \fboxsep 定义的，其默认值是 3pt。这四种命令是：

\shadowbox　边框带阴影的盒子

同 \fbox 一样，边框线的粗细由 \fboxrule 定义，阴影的宽度由长度参数 \shadowsize 定义，默认值为 4pt。

\doublebox　有双层边框的盒子

默认的情况下，内边框线和外边框线的粗细分别是 0.75\fboxrule 和 1.5\fboxrule plue 0.5pt。

\ovalbox　圆角边框盒子

对于这种盒子，圆角边框线的粗细是由命令 \thinlines 定义的，圆角弧直径的默认定义为 \cornersize{0.5}，它将圆角弧的直径定义为盒子的长和宽中较短的那个乘以 0.5。另外，带星号的命令 \cornersize*{*dim*} 将圆角弧的直径设置为数 *dim*，但这实际上是近似的，因为 LaTeX 对圆弧的尺寸是有限制的。

\Ovalbox 粗线圆角边框盒子

这种圆角边框盒子与由 \ovalbox 定义的类似，只是边框线的粗细由命令 \thicklines 来定义。

fancybox 宏包还定义了一个 Sbox 环境，其功能相当于标准 LaTeX 中的 \sbox 命令和 LaTeX2e 中的 lrbox 环境，它先将环境中的内容储存起来，然后用命令 \TheSbox 调用。但要注意 Sbox 环境与 \sbox 命令不同之处是：当使用命令 \TheSbox 调用储存在 Sbox 中的内容时，也同时清除了这个环境中的内容，因而不能多次调用。利用 Sbox 环境，用户可以定义各种有特殊功能的环境。例如下面的命令：

```
\newenvironment{ovalminipage}{\begin{Sbox}\begin{minipage}}%
                {\end{minipage}\end{Sbox}\ovalbox{\TheSbox}}
```

就定义了一个带圆角边框的小页环境。

```
\begin{ovalminipage}{5.8cm}
This is a minipage framed by a
framebox with an arc corner.
\end{ovalminipage}
```

This is a minipage framed by a framebox with an arc corner.

2.15 定义新命令和新环境

虽然 LaTeX 系统自身有丰富的命令和环境，但它也不是万能的。在现有的命令和环境中你可能找不到一个合适的命令或环境来直截了当地排出你想排的结果。你也可能对某个现有的命令或环境的效果不太满意，而想对它进行一些改进。此时，就需要一种方法来定义新命令、新环境或者修改已有的命令、环境。

2.15.1 定义新命令

在 LaTeX2e 系统中，下面这条命令用来定义新的命令：

\newcommand{*name*}[*num*][*default*]{*definition*}

其中 *name* 是新命令的名称，它具有带反斜杠的标准命令形式； *definition* 是对这个新命令的定义； *num* 是一个选项，它是 1 到 10 之间的一个数，表示新命令中的参数个数； *default* 是新命令参数的默认选项，它只赋予第一个参数。下面三个例子可以帮助我们理解如何定义新命令：

```
\newcommand{\companion}
 {The LaTeX Companion} \companion
 \ldots\\ \companion
```

The LaTeX Companion...
The LaTeX Companion

在此例中我们定义了命令 \companion 为 The LaTeX Companion。当我们需要反复输入一段文字时，就可以定义一个简单的命令来代表这段文字。在下例中我们定义了一个带两个参数的命令 \thisthat。定义中的 #1 将被第一个参数替换，而 #2 将被第二个参数替换。

```
\newcommand{\thisthat}[2]{This
one is #1, but that one is #2.}
\thisthat{short}{long}\\
\thisthat{good}{bad}
```

This one is short, but that one is long.
This one is good, but that one is bad.

第三个例子我们定义一个有默认值的命令：

```
\newcommand{\Example}[2][YYY]
{Mandatory arg: #2; Optional arg: #1.}
\Example{BBB}\\
\Example[XXX]{AAA}
```

Mandatory arg: BBB; Optional arg: YYY.
Mandatory arg: AAA; Optional arg: XXX.

注意，用 \newcommand 不能定义一个已经存在的命令，不过我们可以用一条特殊命令 \renewcommand 来对已存在的命令进行更新。\renewcommand 的用法与 \newcommand 的用法完全相同。

另外，我们也可以用 \providecommand 来定义新命令，其用法也与 \newcommand 相同。如果在 \providecommand 中定义的命令已经存在，则这个定义不起作用，也无错误信息。

2.15.2 定义新环境

类似于 \newcommand，也有一条命令 \newenvironment 可以用来定义新环境。这条命令的用法如下：

\newenvironment{*name*}[*num*][*default*]{*beg-def*}{*end-def*}

其中 *name* 是新环境的名称； *num* 是一个选项，它是 1 到 10 之间的一个数，表示新环境中的参数个数； *default* 是新环境的默认参数选项，它只赋予第一个参数； *beg-def* 是环境的开始定义，当系统读到 \begin{name} 时就调用它； *end-def* 是环境的结束定义，它将替代 \end{name}，也就是说，新环境中的内容实际被放在 *beg-def* 和 *end-def* 中。注意，在定义中代替参数值的 #1，#2 等不能出现在结束定义*end-def* 中。例如我们可以

用下面的代码：

```
\newsavebox{\Fminibox} \newlength{\Fminilength}
\newenvironment{Fminipage}[1][\linewidth]%
{\setlength{\Fminilength}{#1-2\fboxsep-2\fboxrule}%
\begin{lrbox}{\Fminibox}\begin{minipage}[s]{\Fminilength}}%
{\end{minipage}\end{lrbox}\noindent\fbox{\usebox{\Fminibox}}}
```

定义一个带边框的小页环境，边框的默认宽度是正文主体的宽度。注意，因为在定义中我们利用了宏包 calc 的关于参数中的长度可用算式替代的优点，所以必须使用宏包 calc 才能使用这个环境。

```
\begin{Fminipage}[6cm]
Here the text is placed in a
minipage with a frame box.
\end{Fminipage}
```

Here the text is placed in a minipage with a frame box.

2.15.3 命令的作用范围

在源文件的导言部分中定义的命令或环境对整个文件都起作用，但是在某环境中定义的新命令或新环境只在该环境中起作用，离开了那个环境，所定义的东西将不起作用，而且想重新定义时也只能用 \newcommand 或 \newenvironment 而不是用 \renew 的形式。

对于那些起整体作用的命令或环境，即那些在文件的导言部分中定义的命令或环境，要重新定义时必须用 \renewcommand 或 \renewenvironment。然而如果这种重新定义是在某个环境中进行的，则重新定义的东西也就只能在这个环境中有效了，离开了这个环境以前的定义仍起作用。

对于嵌套的环境也有类似的情况。外层环境中的定义在所有内层环境中也有效，因而在内层环境中也只能用 \renewcommand 或 \renewenvironment 来重新定义，而且一离开这个内层环境，重新定义的内容就不起作用了，而旧的定义又恢复了作用。

要注意的是所定义的结构如果被 \newsavebox 或 \newcounter 保存起来，那么它们就有了整体作用，不管它们起初是在哪个环境中定义的，离开了这个环境也起作用。

2.15.4 定义的顺序

在自定义结构的内部可以调用其他的命令。在通常情况下，处于自定义结构内层的被调用的命令在进入该自定义结构之前就已经定义好了，但这是不必要的，内部被调用的命令也可以在自定义结构之后再来定义。但非常重要的一点是使用其他的命令来调用

一个自定义结构时，这个自定义结构必须在这之前已经定义好了。

下面是一个嵌套定义的例子：

```
\newcommand{\A}{结构A}
\newcommand{\B}{结构B}
\newcommand{\C}{\A \B}
```

在这个嵌套定义中，命令 \C 的定义中包含了以前定义的命令 \A 和 命令 \B。这个嵌套定义也可以换一个写法：

```
\newcommand{\C}{\A \B}
\newcommand{\B}{结构B}
\newcommand{\A}{结构A}
```

2.15.5 参数的传递

在定义命令或环境时，所定义的命令或环境的哑参数变量，即 #1, #2 等可以作为这个定义中的其他命令的参数。比如当 \A 和 \B 分别是两条含有一个参数的命令，那么下面的定义

```
\newcommand{\C}[3]{\A{#1}#2\B{#3}}
```

是合法的，它定义了一条含有三个参数的命令 \C，第一个和第三个参数分别传给了命令 \A 和 \B, 只有第二个参数被它自己直接使用。直接参数和传递参数之间也允许各种各样的组合。下面是一个更复杂的例子：

```
\newcommand{\anvec}[2]{\mbox{%
$(#1_1,\cdots,#1_#2)$}}
\newcommand{\sumvec}[4]{\anvec{#1}{#4}%
=\anvec{#2}{#4}+\anvec{#3}{#4}}
$$\sumvec{z}{x}{y}{n}$$
```

$$(z_1,\cdots,z_n)=(x_1,\cdots,x_n)+(y_1,\cdots,y_n)$$

2.15.6 定义的嵌套

自定义结构是可以嵌套使用的，也就是说在自定义结构中还可以定义另一个结构。因而下面的结构：

```
\newcommand{\outer}{{\newcommand{\inner}{…}}}
```

是合法的。根据对定义的作用范围的解释，内部定义的命令 \inner 只在外部命令 \outer 中有效。虽然这种嵌套的自定义结构可以限制临时命令被保存的时间，从而不占有过多的内存，但还是建议不要过多的使用，这是因为嵌套定义结构中有很多括号，其中不同结构之间的关系不是十分明了，一旦在定义中遗漏了一对括号，就将导致在第

二次调用此定义时出现错误。下面是一个例子，从中可体会嵌套定义的复杂性。

```
\newcommand{\ninelove}{{\newcommand{%
\threelove}{{\newcommand{\onelove}{%
I love LaTeX!}\onelove\ \onelove\
\onelove}} \threelove\\ \threelove\\
\threelove}}
My opinion of LaTeX:\\[0.5ex]\ninelove
```

My opinion of LaTeX:
I love LaTeX! I love LaTeX! I love LaTeX!
I love LaTeX! I love LaTeX! I love LaTeX!
I love LaTeX! I love LaTeX! I love LaTeX!

如果内部定义和外部定义都含有参数，那么内部定义的哑参数符号和外部定义的哑参数符号必须有区别。LaTeX 规定最外层定义的哑参数符号用 #1,···,#9 表示，第二层内部定义的哑参数符号用 ##1,···,##9 表示。下面是一个外层定义和内层定义都含有参数的例子：

```
\newcommand{\thing}[1]{{\newcommand%
{\thecolor}[2]{The ##1 is ##2.}
\thecolor{#1}{red} \thecolor{#1}{green}}}
The color of the objects are\\[0.5ex]
\thing{dress}\\ \thing{book}\\
\thing{car}
```

The color of the objects are
The dress is red. The dress is green.
The book is red. The book is green.
The car is red. The car is green.

这个例子中的定义也可以用下面较容易理解的方法来写：

```
\newcommand{\thing}[1]{\thecolor{#1}{red} \thecolor{#1}{green}}
\newcommand{\thecolor}[2]{The #1 is #2.}
```

2.15.7 多余的空白

在使用自定义结构时，有时会产生一些多余的空白，这种情况通常是由于自定义结构中添加了不必要的空格或换行而造成的。这样做可能仅仅是为了使源文件便于阅读，而定义中恰恰不允许。从前面的许多例子的定义中我们看到很多定义的行末都使用了“%”这个符号，这正是为了避免出现多余空格的缘故，因为 LaTeX 将换行符解释为一个空格。

在 LaTeX 中有许多不可见命令（invisible command），也就是说在这类命令出现处不会输出任何可见符号。如果这类不可见命令的周围是空格的话，那么在它们出现的地方可能输出两个空格。例如：

```
For example \rule{0pt}{0pt} produces
```

将输出“For example produces”。在 example 和 produces 之间出现了两个空格。不带参数的不可见命令就不会产生这种问题，因为这类命令实际上是以空格符作为命令结

束符的。另外，下面的 LaTeX 命令和环境如同“%”也总是将其后面处于同一行中的空格消除。

```
\pagebreak      \linebreak      \label      \glossary      \vspace
\nopagebreak    \nolinebreak    \index      \marginpar
figure          table
```

2.16 控制结构语句 —— ifthen 宏包

任何计算机编程都离不开条件控制语句。虽然 LaTeX 是排版系统，但在使用 LaTeX 排版文稿时，有时也希望同一排版对象在不同的情况下具有不同的排版效果，此时条件控制语句就非常有用了。ifthen 就是一个提供了条件控制语句的宏包，使得用户在使用 LaTeX 时能建立控制结构。

`\ifthenelse{判断条件}{肯定结构}{否定结构}`

这条命令指出，当判断条件为真时执行肯定结构，判断条件为假时执行否定结构。其中判断条件可以是如下的一些结构：

`前数字 < 后数字`：判断前数字是否小于后数字。

`前数字 = 后数字`：判断前数字是否等于后数字。

`前数字 > 后数字`：判断前数字是否大于后数字。

`\isodd{数字}`：判断数字是否是奇数。

`\equal{前字符串}{后字符串}`：判断前字符串是否与后字符串匹配。

`\lengthtest{前度量 < 后度量}`：判断前度量是否小于后度量。

`\lengthtest{前度量 = 后度量}`：判断前度量是否等于后度量。

`\lengthtest{前度量 > 后度量}`：判断前度量是否大于后度量。

`\boolean{布尔参数名}`：判断布尔参数的值是否为真。

例如：

```
\newcommand{\oddeven}[1]{Number #1 is %
  \ifthenelse{\isodd{#1}}{odd}{even}}
 \oddeven{4}.\\ \oddeven{5}.
```

Number 4 is even.
Number 5 is odd.

`\whiledo{判断条件}{`*while* `语句}`

这条命令指出只要当判断条件为真时就不断地执行 *while* 语句，直到判断条件为假。这里判断条件的用法与上面相同。注意，使用这条命令时，在 *while* 语句中一定要有一种

机制使判断条件能转化为假，否则语句将一直执行下去，而不停止。这就得不到输出结果了。下面是一个结合 \whiledo、\stepcounter 和 \value 的例子：

```
\newcounter{MYctr}
\whiledo{\value{MYctr} < 5}%
{你会喜欢用~LaTeX~的。 %
\stepcounter{MYctr}}
```

你会喜欢用 LaTeX 的。你会喜欢用 LaTeX 的。你会喜欢用 LaTeX 的。你会喜欢用 LaTeX 的。你会喜欢用 LaTeX 的。

在这个例子中，每执行一次语句，计数器 MYctr 的值就增加 1。因此执行 5 次后 MYctr 的值是 6，从而终止执行语句。利用 ifthen 宏包以及下节介绍的 calc 宏包可以在 LaTeX 中做相当复杂的计算。

2.17 算术运算 —— calc 宏包

TeX 中的算术运算通常是由像 \advance 和 \multiply 这样的底层命令来完成的。这类命令一般只是用来开发新宏包，难于为普通用户所使用。calc 宏包定义的几条命令增强了 LaTeX 的算术运算功能，而且这些命令非常容易理解，使用也很方便。它使我们可以直接处理包含整型和度量的表达式，而不仅仅是整数和度量本身。

一个整型表达式是一个表达式，它只包含整数、TeX 寄存器、计数器、圆括号和二元算术运算符（+, -, *, /）。例如，我们用加号就可以让计数器的值增 1，而无需用 \stepcounter 命令：

```
\newcounter{Local}\setcounter{Local}{2}
计数器~Local~的原始值是~\theLocal.
\setcounter{Local}{\value{Local}+1}
现在计数器~Local~的值是~\theLocal.
```

计数器 Local 的原始值是 2. 现在计数器 Local 的值是 3.

在 TeX 系统中，\time 是个寄存器，表示从午夜零点到现在有多少分钟。下面利用它并结合 calc 宏包来定义一个打印当前时间的命令：

```
\newcounter{hours}\newcounter{minutes}
\newcommand{\printtime}{\setcounter{hours}{\time/60}%
\setcounter{minutes}{\time-\value{hours}*60}%
\thehours~点~\theminutes~分}

现在的时间是~\printtime。
```

现在的时间是 13 点 25 分。

当涉及长度参数时，同一个表达式中用于加法和减法的量必须是同型的。像“2cm+4”这样的表达式是不允许的，而“2cm+4pt”是合法的，因为 2cm 和 4pt 都

是长度。对于乘法和除法来说，乘数和除数必须是整数，因此“2m*4”是合法的，而“2m*4pt”却是不合法的。另外，长度参数必须放在前面，而乘数和除数只能在后面，因此，不可以用“4*2cm”这样的表达式。注意除法所得的结果都取整数部分。

当不得不用一般的十进制数作除数或乘数时，必须把它转换成整型的数。

```
\real{十进制数}
```

就是一个将十进制数转换成整型数的命令。而命令

```
\ratio{长度表达式}{长度表达式}
```

计算两个长度的比值，并输出整型数。例如：

```
\setcounter{Local}{7/2} \theLocal,
\setcounter{Local}{3*\real{1.6}}
\theLocal,                                    3, 4, 5,
\setcounter{Local}{3*\ratio{7pt}{4pt}}%
\theLocal,
```

最后再举一个例子。假设有一幅图其原始宽度和高度分别由 \Xsize 和 \Ysize 表示，现在要把它伸缩到使其宽度与整个页面的宽度一样，并且保持宽度与高度的比值，那么可以用命令

```
\setlength{\newYsize}{\Ysize*\ratio{\textwidth}{\Xsize}}
```

来设置伸缩之后的图形高度 \newYsize。

第 3 章　文章结构及排版

在前面的章节中我们简要地阐述了 TeX 和 LaTeX 的发展历史、 LaTeX2e 源文件的整体结构以及 LaTeX2e 的类型和宏包等概念，在本章及以后的各章我们来讨论编写源文件的一些细节问题，以使我们能排版出满意的文稿。

3.1　文章和语言的结构

写文章的目的是要向读者传达一种思想、信息或者知识。如果这种思想、信息或者知识有清晰明了的结构，那么读者便容易理解；如果作者写出的文章的书面结构也能够与作者的思想结构一致起来反映出逻辑上和语义上承转变化，那么作者的思想就更容易被读者理解了。

LaTeX 不同于其他一些排版系统，你只需告诉它文章的逻辑和语义结构，它就会根据源文件的类型和一些宏包中定义的“规则”排印出文章的书面结构。

在 LaTeX 文件中最重要的文本单位是段落，因为一个段落就是一个具有连续语义的书面形式。在后面几节中我们要学习如何用 \\ 来强行断行和用一条空行来分段。

另一个较小的文本单位是语句。在英语中是用一个点加上一个稍大些的空白来结束一条语句的。然而在缩略语后面通常也跟有一个点，因而 LaTeX 有时分不清这个点是语句的结束还是缩略语的结束，这就需要我们在源文件中指明。

3.2　字距、行距和段落间距

3.2.1　字距

为了使版面整齐，便于阅读， LaTeX 会在字与字之间插入不同的空白，并且在句末插入一个较大的空白。这些空白会随着字体的改变而改变。通常 LaTeX 是以句号（一个点）、问号和感叹号作为句子的结束符号。但若一个点跟在一个大写字母后面， LaTeX 并不认为这是一个句子的结束，因为通常的缩略语就是大写字母后面加一个点。

一个反斜杠 \ 后面加一个空格就会输出一个通常的空格，它不随字体变化而改变。一个波浪号“~”也输出一个不能增大的空白，而且也不能在波浪号之处换行。因此波浪号经常用在姓名之中。一个点前面若有命令“\@”则表示此点是一个句号，无论这个点前面是大写字母还是小写字母。请看下例：

```
Mr.~Smith was happy to see her\\
cf.~Fig.~5\\
I like BASIC\@. What about you?
```

Mr. Smith was happy to see her
cf. Fig. 5
I like BASIC. What about you?

如果不想在句号后面跟一个较大些的空白，则可以使用命令 \frenchspacing。这条命令通常用于非英语的 LaTeX 源文件中，它通知 LaTeX 使句末的空白与普通词之间的空白保持一致。若用了 \frenchspacing 命令，\@ 就不必再用了。

由 Philip Taylor 编写的 letterspace 宏包（即 lettersp.sty）定义了一条用来改变字母之间和词语之间距离的命令 \letterspace，其语法如下：

\letterspace [介词] [*width*]{*some text*}

其中的 *width* 是一个确切的长度（如 4cm, \textwidth 等）。介词是 to 或 spread。当介词是 to 时，命令表示创建一个宽为 *width* 的水平盒子并将 *some text* 在盒子中伸展开来充满此盒子。当介词是 spread 时，命令表示创建一个盒子，其宽度为 *some text* 的自然宽度加上 *width*，然后将 *some text* 在盒子中伸展开来并充满盒子。此宏包中还定义了命令 \naturalwidth 表示自然宽度。例如：

```
\letterspace to .9\naturalwidth%
          {Time for good men}
\letterspace {Time for good men}
\letterspace to \textwidth%
         {Time for good men}
\letterspace to .5\textwidth%
         {Time for good men}
\letterspace spread 0.5\naturalwidth%
          {Time for good men}
\letterspace to 1.5\naturalwidth%
          {Time for good men}
```

Time for good men
Time for good men
T i m e f o r g o o d m e n
Time for good men
T i m e f o r g o o d m e n
T i m e f o r g o o d m e n

在使用 \letterspace 时应当注意此命令总是创建一个水平盒子，所以不要将盒子的实际宽度设置的太大以免超过页面的宽度。另外命令中括号后面的文字会被排到下一段去。实际上此命令只是常被用于标题中，这是因为对于大字体的标题，字母间距大一点会显得好看些。此命令也不支持中文。

3.2.2 行距

如果要改变整个文档的行距，可以在源文件的导言部分使用 \linespread 命令，语法为

\linespread{*factor*}

其中 *factor* 是一个因子，它是一个十进制小数，代表普通行距的倍数。通常情况下行距没有增大，因此这个因子的默认值是 1。\linespread{1.3} 表示通常所说的“一个半”行距，而 \linespread{1.6} 表示“双倍”行距。

另外，也可以设置命令 \baselinestretch 来改变整文的行距：

\renewcommand{\baselinestretch}{*factor*}

其中 *factor* 也是一个因子，意义同上。如果将上述两条命令置于主体部分，那么它们只改变命令之后的行距。

在主体部分中也可以使用命令

\baselineskip=*length*

来将命令之后的行距设置为 *length*, 这是一个长度，如 5pt、8mm 等。但 \baseineskip 的值会随着不同字体的引入而改变。这是因为当使用一个新字体时，会同时赋予 \baseineskip 一个新的值。注意，这条命令不能用于导言部分。

3.2.3 段落间距和首行缩格

在 LaTeX 中，放置在导言部分的下述两条命令

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

将影响整篇文本的段落格式。其中包含 \parindent 的第一条命令将段落的首行缩格改为零，而包含 \parskip 的第二条命令增大了段落间距。在欧洲大陆经常可以看到这种文本格式。注意，这种设置对目录也有影响，它使目录的行距看起来很稀松。要避免这种情况你也许想把这两条命令移到排版目录的命令 \tableofcotents 之后的某个地方，或者干脆不用它们。事实上，许多专业排版的书籍都使用了段落首行缩格，而且并不增加段落间距。

在一个首行不缩格的段落的开头使用命令

\indent

可以改变这个段落为首行缩格的。当然这条命令只有在导言部分没有设置段落缩格为零时才有效。另外，按照欧美的传统习惯 LaTeX 的默认设置是把小节的第一个段落设置为不缩格的，要改变这个设置只需在导言部分调用 LaTeX2e 工具宏包套件中的宏包 indentfirst（在 DOS 中名字为 indenrst.sty）。

要使一个段落首行不缩格可以在此段落的开头加上 \noindent 这条命令。

3.3 换行和换段

3.3.1 段落调整

一般来说在书籍或文章的排版中一个段落总是行行对齐的。LaTeX 根据整个段落的情况选择最优的字距和断行处来使每一行对齐，如果有必要它还会在合适的地方将一个词分开并用连字符连接。LaTeX 如何排版段落还依赖于所用的类和宏包。基本上，一个段落的第一行采用缩格的方式，而且段落间距与行距是一样的。

在某些特殊场合，也许有必要通知 LaTeX 在某处立即换行。下面的命令

`\\` 或 `\newline`

会结束当前行并开始一个新行，但不是开始新段。命令 `\\*` 表示换行，但是不能在换行后开始一个新页。而命令

`\newpage`

则表明换行之后开始一个新页。另外，下面这条命令

`\\[`*length*`]`

可以使换行之后附加一段垂直空白，其中 *length* 就是垂直空白的长度。例如：

```
The optional argument
\textit{length} is\\[3mm] a length
that specifies how much additional
line spacing is to be put between
the lines.
```

The optional argument *length* is

a length that specifies how much additional line spacing is to be put between the lines.

下面四条命令

`\linebreak[`n`]`, `\nolinebreak[`n`]`, `\pagebreak[`n`]` 和 `\nopagebreak[`n`]`

的含义犹如其名，分别表示强行断行、无强行断行、强行断页和不强行断页，其中的 n 是 0 到 4 之间的整数，表示命令的强度，数字越大则命令强度越大，n 的默认值是 4。这四条命令在默认值的情况下，表示命令一定要被执行，而在 n 的值小于 4 的情形下，则把命令是否被执行交给 LaTeX 系统来判断。当排版的结果很不好看时命令可能不被执行。另一方面，命令 `\linebreak` 中的 break 与命令 `\newline` 中的 new 的含义是不同的。命令 `\newline` 之后 LaTeX 以空白填满剩余空间，而当命令 `\linebreak` 被执行后，命令所在行仍将被命令前的字符占满，使这一行与别的行对齐，只不过这一行中词

与词间的空白加大了。因此，如果确实想在某处换行，最好是用命令 \newline。请看下例：

```
Do not confuse\linebreak these
``break'' commands with the ``new''
commands. Even when you give a
``break'' command, LaTeX still
tries to even out the right border
of the page.
```

Do not confuse these "break" commands with the "new" commands. Even when you give a "break" command, LaTeX still tries to even out the right border of the page.

LaTeX 总是试图在一个最佳的位置换行。如果它按照高标准找不到这样的最佳位置，那么就会让某一行的字符超过这一段落的右边界，并且在编译源文件时抱怨文件中第几行产生一个“溢出的水平盒子” (overful hbox)。这种情况经常发生在 LaTeX 按照语言的拼读规则找不到插入连字符的合适位置的时候。此时，我们可以使用命令 \sloppy 引导 LaTeX 降低排版的标准从而避免出现溢出的水平盒子，但某些词之间的距离可能被加大了，因而又出现“未满的水平盒子” (underful hbox)，在这种情况下，排版的结果看起来通常也不是很好。另一个相反的命令是 \fussy，它关闭前面的 \sloppy 命令，允许在后面出现溢出的水平盒子。\fussy 命令也是 LaTeX 的默认设置。sloppy 和 fussy 都可当成环境来使用，即，可以利用下面两个环境

```
\begin{sloppy} 段落文本 \end{sloppy}
\begin{fussy} 段落文本 \end{fussy}
```

来调整一个段落。

3.3.2 词语分割

为了得到满意的输出结果，在必要的时候 LaTeX 会将一个词从某处分开插入连字符之后换行。如果插入连字符的算法找不到一个合适的位置，我们可以在源文件中加入一条命令，引导 LaTeX 对一些词进行分割。下述命令

```
\hyphenation{一列词}
```

告诉 LaTeX 系统命令中的一列词应如何分割。注意这一列词中的每个词在允许分割处都应插入一个连字符“-”，若没有连字符则表明该词不能分割，而且每个词都只能包含通常的字母，不能包含其他的特殊符号。这条命令只能用于源文件的导言部分，并且不区分大小写。例如，下面的命令

```
\hyphenation{FORTRAN hy-phen-a-tion}
```

将通知 LaTeX 系统不对 FORTRAN、Fortran 及 fortran 等进行分割，对 hyphenation

及 Hyphenation 等也只能在插入连字符的地方分割。

另外一个更直接的方法是不用命令 \hyphenation，而是将命令 \- 直接插入单词中允许分割处。这条命令对于分割那些包含特殊符号的单词非常有用，因为 LaTeX 并不对包含特殊符号的单词进行分割。例如：

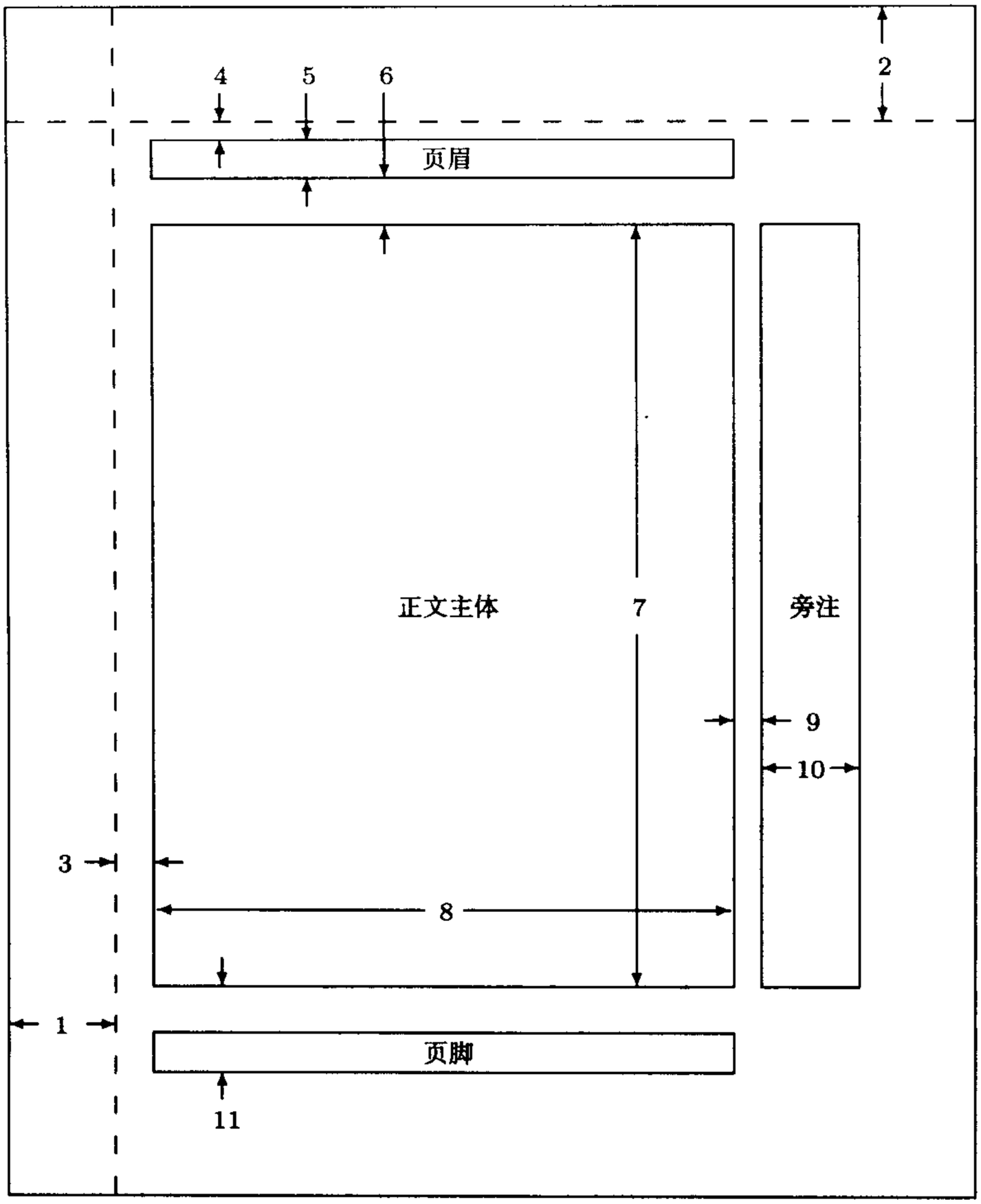
```
Here is a very long word: su\-%
per\-cal\-i\-frag\-i\-lis\-tic\-%
ex\-pi\-al\-i\-do\-cious
```

Here is a very long word: supercalifragilistic-expialidocious

3.4 页 面 尺 寸

在 LaTeX2e 中我们可以通过命令 \documentclass 的参数选项来改变页面的尺寸，例如，可以选 a4paper、a5paper 等。一旦取定了这些选项，LaTeX2e 就会自动设置正文主体 (body) 的高度和宽度以及页眉 (header) 和页脚 (footer) 的高度等。虽然这些自动设置的尺寸合乎大多数人的习惯，但是在某些场合我们仍需要改变它们。图 3.1 之中显示了一部分页面尺寸的参数。下面列举了控制页面尺寸的参数及其含义：

\textheight	正文主体的高度（不包括页眉和页脚）。
\textwidth	正文主体的宽度。
\columnsep	多栏排印格式中，两栏之间的空白的宽度。
\columnseprule	多栏排印格式中，用于分割两栏的垂直线的宽度。
\columnwidth	这条命令用来表示多栏排印格式中，一栏的宽度。 LaTeX 根据 \textwidth 和 \columnsep 会计算出适当的值。
\linewidth	当前行的行宽，通常等于 \columnwidth，但在改变了边距的环境中也许有不同的值。
\evensidemargin	双面排印格式中，双数页左侧附加空白的宽度。
\oddsidemargin	双面排印格式中，奇数页左侧附加空白的宽度，也是所有页左侧附加空白的宽度。
\footskip	页脚基线与正文最后一行的基线之间的距离。
\headheight	页眉的高度。
\headsep	页眉与正文主体之间的距离。
\topmargin	页眉顶部附加垂直空白的高度。
\marginparpush	两个旁注间的最小垂直距离。
\marginparsep	旁注与正文主体之间的水平距离。
\marginparwidth	旁注的宽度。



1	one inch + **\hoffset**	2	one inch + **\voffset**
3	**\oddsidemargin** = 25pt	4	**\topmargin** = 6pt
5	**\headheight** = 15pt	6	**\headsep** = 18pt
7	**\textheight** = 591pt	8	**\textwidth** = 398pt
9	**\marginparsep** = 7pt	10	**\marginparwidth** = 51pt
11	**\footskip** = 23pt		

图 3.1 页面尺寸参数

LaTeX2e 提供了两条命令来改变这些参数的设置，这两条命令只能用于源文件的导言部分。第一条命令是

\setlength{*parameter*}{*length*}

它可以给这些参数设置一个固定的值，其中的 *parameter* 就是上面列举的参数，而 *length* 则是给这个参数赋的值。第二条命令是

\addtolength{*parameter*}{*length*}

它可以把 *length* 这个值加到参数 *parameter* 原先的值上。例如，要把正文主体的高度设置为 21 厘米并且把正文主体的宽度设置成默认值加上 1 厘米，可在前置部分放置下面两条命令：

```
\setlength{\textheight}{21cm}
\addtolength{\textwidth}{1cm}
```

在这里我们提一下 LaTeX2e 工具宏包套件中的宏包 calc，利用它可以用算术运算来代替 \setlength 的选项 *length* 以及其他可以是数字的参数选项。

3.5 页 版 式

3.5.1 标准页版式

页面尺寸中的参数决定之后，每一页正文主体的高度和宽度以及页眉页脚的高度等度量都是不变的，但每一页的页眉和页脚的内容却可以不同。我们可用页版式的设置来决定这些内容。下面两条命令

\pagestyle{*style*} 和 \thispagestyle{*style*}

用来选择页版式 *style*，其中第一条命令用来设置当前页以及后续页的格式，而第二条命令只用来设置当前页的页版式。

标准 LaTeX 有以下四种页版式：

empty	在这种页版式中页眉和页脚都是空的。
plain	页眉是空的，页脚只包含放在中间的页码。
headings	此格式中，页眉包含由文件类型决定的信息，通常是章节名及页码，而页脚是空的。
myheadings	类似于 headings，但页眉的内容可以由用户自己定义。

前两种页版式通常用于标准的文件类型。例如 LaTeX 标准类型会在独立的标题页引用 \thispagestyle{empty}，而在新的部分（命令 \part）或章（命令

\chapter）的第一页会自动引用命令 \thispagestyle{plain}。因而即使在导言部分用了 \thispagestyle{empty}，在每章的第一页仍然会有页码。要避免这种情况，就必须在每一章开始时，也就是在每条命令 \chapter 之后都要使用命令

```
\thispagestyle{empty}
```

或者在导言部分重新定义 plain 页版式，使它于 empty 页版式相同。

在 headings 页版式和 myheadings 页版式中，页眉的内容是由下述两条命令

\markboth{*left_head*}{*right_head*} 和 \markright{*right_head*}

决定的，其中 *left_head* 是左侧页眉内容， *right_head* 是右侧页眉内容。

命令 \markboth 用于双面排印的文件中。通常假定双数页在左侧而单数页在右侧，并且页码排在左页页眉的左侧和右页页眉的右侧。对于单面排印的文件，所有的页都被当作右页，因而只需用 \markright 来决定页眉的内容。同时这条命令也用于双面排印的文件中来覆盖 \markboth 中的 *right_head*。

表 3.1 中展示了 book，report 和 article 这三种文件类型的页眉分别在双面排印和单面排印中的内容。

表 3.1 LaTeX 中的页版式命令

排印方式	命　令	文　件　类　型	
		book, report	article
双面排印	\markboth	\chapter	\section
	\markright	\section	\subsection
单面排印	\markright	\chapter	\section

3.5.2 定义新页版式

虽然前面所述几种标准页版式非常有用，但用户有时仍觉不够，此时可以写一个宏包来定义新的页版式。页版式是由命令 \ps@xxx 来定义的，其中 xxx 就是页版式的名称。当在源文件中使用 \pagestyle{xxx} 时，页版式 xxx 就被启用了。在 \ps@xxx 之后还必须定义下面几条命令，它们包含所定义的页版式的页眉和页脚的内容：

\@oddhead　双面排印中奇数页（或单面排印中的所有页）的页眉。

\@evenhead　双面排印中偶数页的页眉。

\@oddfoot　双面排印中奇数页（或单面排印中的所有页）的页脚。

\@evenfoot　双面排印中偶数页的页脚。

要注意这些命令并非用户命令，而是 LaTeX 系统的“变量”。由于命令中带有符号

@，它们只能用在宏包中，或者被夹在命令 \makeatletter 和 \makeatother 之中。另外，要使章节名以不同的形式出现在页眉中还必须对 \chaptermark 和 \sectionmark 这两条命令重新定义。定义的方法如下：

```
\renewcommand{\chaptermark}[1]{\markboth{
                \chaptername\ \thechapter. #1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection. #1}}
```

其中命令 \chaptername 表示章名，通常印出 Chapter，命令 \thechapter 印出章号，命令 \thesection 印出节号。在 3.17.2 节中，我们将介绍一个很容易定义和修改页版式的宏包。

3.5.3 页码

对于中短篇幅的文稿一般没有必要改变页版式，只要使用文件类型提供的页版式就行了。但是对于长篇文稿，如书籍，就应当考虑排版传统。通常在正文之前的部分，如序言和目录部分，采用罗马数字来标页号，而在主体部分用阿拉伯数字标页号。习惯上，新的一章都是从右页（奇数页）开始的。

页码是由下面这条命令

\pagenumbering{*num_stype*}

来控制的，其中 *num_stype* 是页码格式。标准的页码格式有以下几种：

Alph	大写英文字母 A, B, C,$\cdots$
alph	小写英文字母 a, b, c,$\cdots$
Roman	大写罗马数字 I, II, III,$\cdots$
roman	小写罗马数字 i, ii, iii,$\cdots$
arabic	阿拉伯数字 1, 2, 3,$\cdots$

一般地，命令 \pagenumbering{arabic} 把页码设置成阿拉伯数字并从 1 开始计数，如果想从某一页开始，可以用命令

\setcounter{page}{*page_num*}

它把页码设置成从 *page_num* 开始，其中 \setcounter 是用户修改 LaTeX 内部计数器的命令。

3.5.4 使用 fancyhdr 宏包排印页眉和页脚

在这一小节我们介绍由 Piet van Oostrum 编写的 fancyhdr 宏包，它使我们能够很容易就排印出满意的页眉和页脚。这个宏包有下面几个主要特点：

- 可排印左中右三部分页眉和页脚。
- 可以调整页眉线和页脚线。
- 页眉和页脚的长度可超过正文的宽度。
- 可排印多行页眉和页脚。
- 奇数页和偶数页的页眉和页脚可分别定义。
- 各章节的页眉和页脚可以不同。

此外，还可以完全控制页眉和页脚的字体。要使用这个宏包，首先要确定系统中已安装了文件 fancyhdr.sty，然后在源文件的导言部分输入命令

```
\usepackage{fancyhdr} \pagestyle{fancy}
```

接着根据宏包提供的下述命令

```
\lhead{左边页眉}  \chead{中间页眉}  \rhead{右边页眉}
\lfoot{左边页脚}  \cfoot{中间页脚}  \rfoot{右边页脚}
```

来输入单面排印的页眉和页脚的内容。页眉和页脚中可插入 \\ 使之产生多行的页眉或页脚，但此时必须增加页眉高度(\headheight)和页脚的高度(\footskip)，否则会产生纵向溢出。

对于双面排印的文稿可用带选项的命令

```
\fancyhead[选项]{页眉内容}  \fancyfoot[选项]{页脚内容}
```

来输入页眉和页脚的内容，其中的选项是字母 E,O,L,C,R 和它们的组合，字母组合的顺序无关紧要。表 3.2 中列出了这些选项及其含义。例如 [RO,LE] 表示奇数页右边以及偶数页左边，[LO,CE] 表示奇数页左边以及偶数页中间，等等。如果不给出 E 或 O，则表明定义的内容适合所有页。事实上，命令 \lhead 就是命令 \fancyhead[L] 的简化。选项中的字母也可以是小写字母。

表 3.2 fancyhdr 的选项和含义

选项	E	O	L	C	R
含义	偶数页	奇数页	左边	中间	右边

页眉线和页脚线的粗细可用命令

```
\renewcommand{\headrulewidth}{length}
\renewcommand{\footrulewidth}{length}
```

来定义，其中 length 是一个长度，默认值是 0.4pt。

页眉线和页脚线的形式是由 \headrule 和 \footrule 这两个参数控制的，可以更改这两个参数从而产生不同风格的页眉线和页脚线。例如，使用命令

```
\renewcommand{\headrule}{\hrule height0pt%
\vbox to 0pt{\hbox to\headwidth{\dotfill}\vss}}
```

可把实线更改为点线。用下面的命令

```
\renewcommand{\headrule}{%
\hrule height0.8pt width\headwidth \vskip0.8pt %
\hrule height0.4pt width\headwidth \vskip-2pt}
```

可以将页眉线改为两条水平直线，第一条粗 0.8pt，第二条粗 0.4pt，它们之间的距离为 0.8pt。减去 1.6pt 是为了抵消第一条线及两条线之间的间隔所占的垂直距离从而使整个页面的高度保持不变。结合宏包 ulem 提供的画波浪线的命令，还可以设计出直线与波浪线相结合的页眉线。例如，下面的命令

```
\renewcommand{\headrule}{\vskip-1pt\hrule height0pt width\headwidth%
\uwave{\makebox[\widthwidth]{}} \vskip-2.5pt}
```

将页眉线变为单独一条波浪线，而命令

```
\renewcommand{\headrule}{\hrule width\headwidth%
\uwave{\makebox[\linewidth]{}}\vskip-3.9pt}
```

将页眉线设计成水平直线下面加上一条波浪线。

另外，可用命令参数 \footruleskip 来设置页脚线与第一行页脚的距离。

页眉和页脚的长度是由参数 \headwidth 决定的，在默认的情况下它的值等于版心的宽度，即 \textwidth。但可用命令 \setlength 或 \addtolength 来改变它。下面是一个使用 fancyhdr 宏包的例子：

```
\fancyhead{}
\fancyhead[RO,LE]{head}
\fancyfoot[LE,RO]{\thepage}
\fancyfoot[LO,CE]{Grant}
\fancyfoot[CO,RE]{Smith}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}
```

head

正文

51 Grant Smith

3.6 多 栏 排 版

标准的 LaTeX 系统可以对整篇文稿进行双栏排版 (twocolumn) 或单栏排版 (onecolumn)。但它不能对文稿的一部分进行双栏排版而另一部分进行单栏排版，而且在双

栏排版的最后一页，常常是右栏比左栏短，两栏不能平衡。为解决这些问题， Frank Mittelbach 编写了 multicol 宏包（存放在 LaTeX2e 工具宏包集中）专门用于文稿的多栏排版。这个宏包定义了一个具有如下特点的 multicols 环境：

- 排版的栏数可以自由选择（多达 10 栏）。
- multicols 环境中的最后一页的两栏保持平衡，即两栏等长。
- 此环境可用于其他环境之中，例如可用于图形环境和小页环境中。
- 用户可自由定义分栏线的粗细和栏间距的大小。
- 定义多栏格式可作用于整篇文稿，也可只作用于个别环境。

multicols 环境的使用命令是：

```
\begin{multicols}{栏数}[preface ][skip]
多栏排版的文字
\end{multicols}
```

其中的选项 *preface* 是环境中一小段不想进行多栏排版的内容（比如标题等），选项 *skip* 是一个长度，表示当前页的空白小于这个长度时从下一页开始多栏排版。下面是一个两栏排版的例子：

```
\setlength{\multicolsep}{-1mm}
\setlength{\columnseprule}{0.4pt}
\begin{multicols}{2}%
[\section*{About skip}]
The multicols environment starts a new
page if there is not enough free
space left on the current page.
\end{multicols}
```

About skip

The multicols environment starts a new page if there is not enough free space left on the current page.

下面是控制 multicols 环境的参数及其含义，可以用 \setlength 命令来改变这些参数的值，但必须在 multicols 环境之前改变参数的值：

\premulticols　预先检查的环境前的空白长度。页面的剩余垂直空白不小于此长度，或者不小于环境选项 *skip* 所赋予的值时，环境才会立即被执行，否则会先执行 \newpage 命令，再从下一页开始多栏排版。此参数的默认值是 50pt。

\postmulticols　要检查的环境后的空白长度。如环境结束之后页面剩余空白小于此长度，则环境结束时先执行 \newpage 命令，再从下一页开始排版环境后面的文字。此参数的默认值是 20pt。

\multicolsep　放置在环境前后的一段空白距离的长度。此参数的默认是 12pt plus 4pt minus 3pt，这是一个弹性长度。

\columnsep　两栏之间的距离，即栏间距。默认值是 10pt。

\columnseprule　分栏线的粗细。默认值是 0pt，即无分栏线。

在默认的设置下，multicols 环境会通过拉伸各栏之中可以使用的垂直空白，从而使各栏的长度保持相同。如果在环境之前使用命令 \raggedcolumns，那么各栏的底部就允许出现一些多余的空白。如果不要求所有的栏都有同样的长度，而是允许靠左边的栏有更多的文本行，那么可以增大 multicols 环境所使用的计数器 unbalance，这个计数器的数值表示允许右边有多少栏不平衡（即，长度可以不同）。当环境结束时，计数器 unbalance 的值又会被自动设置为零。比如下面的例子中将计数器 unbalance 的值设置为 1，因而允许最后一栏的长度与其他栏的长度不同：

```
\setlength{\columnseprule}{0.4pt}
\begin{multicols}{3}\raggedright
Here is some text to be distributed
over several columns. In this example
ragged right typesetting is used.
\setcounter{unbalance}{1}
\end{multicols}
```

Here is some text to be distributed over several | columns. In this example ragged right typesetting is | used.

3.7 字体的选择

虽然在编写 LaTeX 文件时，系统会根据源文件中的逻辑标识（章、节、脚注等）自动地选择合适的字体。例如对于节名，即 \section 中的字符，系统会在文件类型中定义的主体文字尺寸的基础上用大一号的黑体字将节名排出。但有时用户也有必要根据自己的需要来改变字体的属性。

在 LaTeX2e 中大多数选择字体的命令都有两种形式。一种是带有一个参数的形式，例如 \textbf{···}；另一种是声明形式，例如 \bfseries。声明形式的字体选择命令不带参数，它只引导系统将此命令之后的文字字体改变，直到当前环境结束。我们通常都使用带参数的选择字体命令来改变某些词或语句的字体，而声明形式的选择字体命令一般在定义命令和环境时使用。另外，我们也可以将声明形式的选择字体命令去掉反斜杠之后作为环境来使用，例如：

```
Some words in this sentence are
\begin{bfseries} typeset in bold
\end{bfseries} letters.
```

Some words in this sentence are **typeset in bold** letters.

表 3.3 选择字体的命令

命 令	声 明 形 式	作 用
\textrm{···}	{\rmfamily ···}	roman 罗马字体族
\textsf{···}	{\sffamily ···}	sans serif 字体族
\texttt{···}	{\ttfamily ···}	typewriter 打字机字体族
\textmd{···}	{\mdseries ···}	medium 中等权重序列
\textbf{···}	{\bfseries ···}	**bold** 黑体序列
\textup{···}	{\upshape ···}	upright 直立形状
\textit{···}	{\itshape ···}	*italic* 斜体形状
\textsl{···}	{\slshape ···}	*slanted* 倾斜体形状
\textsc{···}	{\scshape ···}	SMALL CAPS 形状
\emph{···}	{\em ···}	*emphasized* 强调的字体
\textnormal{···}	{\normalfont ···}	文件的主字体

表 3.3 中列出了各种选择字体的命令和它们的声明形式。

当我们要切换到文稿的主体文字时，可以使用命令 \textnormal{···}，或者它的声明形式 \normalfont。在定义新命令和新环境时，若希望这种定义总是用同一种字体而不受周围字体的影响，也可以使用这两条命令。

在 LaTeX2e 中，字体的属性是一个很重要的概念。一般来说字体的属性指的是字体的族 (family)，字体的序列 (series)，字体的形状 (shape)和字体的大小尺寸 (size)。这四种字体的属性是相互独立的，一种属性的改变不会影响字体的其他属性。

3.7.1 字体族

通常情况下，LaTeX2e 中的字体被分成三个族 (family)，它们是 serif 字体族（serif family）、sans serif 字体族 (sans serif family) 和打字机字体族 (typewriter family)。在源文件中，这三种字体族可以分别用 \textrm、\textsf 和 \texttt 这三条命令来调用，而这三种字体命令的声明形式分别是 \rmfamily、\sffamily 和 \ttfamily。大多数的文件类型都是把 serif 字体族作为文件的主要字体，因而命令 \textrm 并不经常使用。不过，如果把文件的主要字体设置成其他字体族，则命令 \textrm 可能会经常出现了。

当使用这些命令时，系统就会调用某些特别的字体，至于调用哪种字体要依赖于所用的文件类型（class）。在默认的安装方式下，serif 字体族使用计算机现代罗

马字体（Computer Modern Roman），sans serif 字体族使用计算机现代 sans 字体（Computer Modern Sans），打字机字体族使用计算机现代打字机字体（Computer Modern Typewriter）。不过可以在导言部分使用选择字体的宏包或者选择字体的命令对之进行改变。

3.7.2 字体序列

字体的序列包含两个方面的因素：宽度 (width) 和权重 (weight)。这里权重指的是字体的黑度。LaTeX2e 提供了两条命令来改变字体的序列：`\textmd` 和 `\textbf`，相应的声明命令是 `\mdseries` 和 `\bfseries`。第一条命令用来选择具有中等宽度值和权重值的字体，而第二条命令则用来选择通常的黑体字。宽度和权重的实际值依赖于文件类型字体大小选项 (10pt,11pt 或 12pt) 以及所用的宏文件。

3.7.3 字体形状

字体的第三种独立的属性是形状 (shape)。大多数文件类型都是采用直立的字体形状作为文件的主要字体形状。这种直立的形状 (upright shape) 可以用命令 `\textup` 或相应的声明形式 `\upshape` 来调用。

另外两种很重要的字体形状是意大利式斜体 (*italic*) 和 SMALL CAPS 字体（一种将小写字母改为较小尺寸的大写字母，不过在字母的粗细上与一般的大写字母稍有不同）。这两种字体可以用 `\textit` 和 `\textsc` 来调用，相应的声明形式分别是 `\itshape` 和 `\scshape`。

作为斜体的另一种选择是所谓倾斜体 (*slanted shape*)，它是将文字在原来字型的基础上向右倾斜一点。可以用 `\textsl` 来选用这种倾斜体，相应的声明形式是`\slshape`。通常一个字体族只包含斜体和倾斜体两种之中的一种，而计算机现代罗马字体族则包含了这两种字体形状。

另外一个与字体形状有关的命令是 `\emph`。这条命令被用来强调一段文字，与它相应的声明形式的命令是 `\em`。一般来说此命令将被强调的语句转成斜体字 (*italic*)，但如果在一段斜体文字中使用这条命令，则被强调的语句就被还原成直立的形状。通常情况下斜体语句的两端与直立文字之间的空白比较小，所以需在斜体语句的两端加上 `\/`。不过使用命令 `\emph` 可以避免这种情况，它会自动在斜体语句的两端加上适当的空白。例如：

```
{\em Nevertheless, one has to be
careful about the\/ {\em proper\/}
use of italic corrections on both
ends of the emphasized text}. It
is therefore better to use the
\verb|\emph| command, which
\emph{automatically} takes care of
the italic correction on both sides.
```

Nevertheless, one has to be careful about the proper *use of italic corrections on both ends of the emphasized text.* It is therefore better to use the **\emph** command, which *automatically* takes care of the italic correction on both sides.

3.7.4 字体的大小

LaTeX 有 10 种改变字体大小的命令，见表 3.4。因为改变字体的命令通常用于定义命令和环境，所以这 10 种命令都是不带参数的声明形式的命令。由于 LaTeX2e 使用了新字体选择方案，字体大小的改变并不影响字体的其他属性，而在 LaTeX2.09 中改变字体的命令将把字体转成文件的主字体。例如：

```
{\small The small and
\textbf{bold} Romans ruled}
{\Large all of great big
\textit{Italy}.}
```

The small and **bold** Romans ruled all of great big *Italy*.

表 3.4 改变字体大小的命令

命　令	结　果	命　令	结　果
\tiny	tiny font	\Large	larger font
\scriptsize	very small font	\LARGE	very large font
\footnotesize	quite small font	\huge	huge
\small	small font	\Huge	largest
\normalsize	normal font		
\large	large font		

不管是 LaTeX2.09 还是 LaTeX2e，与选择字体大小的命令相对应的文字的实际尺寸都依赖于文件的类型的选项 (10pt,11pt 及 12pt) 和所用的宏包。一般来说，命令 **\normalsize** 总把字体切换到文件主字体的大小，字体大小的命令从 **\tiny** 到 **\Huge** 逐渐将尺寸增大。但有时可能两条命令对应着同一个字体尺寸，例如当我们选择 12pt 作为文件主字体的大小时， **\Huge** 与 **\huge** 实际是一样的。表 3.5 中列出了字体选择命令在不同大小的主字体中所对应的实际尺寸。

表 3.5 字体大小的实际点数

主字体大小	10pt (默认值)	11pt (可选值)	12pt (可选值)
\tiny	5pt	6pt	6pt
\scriptsize	7pt	8pt	8pt
\footnotesize	8pt	9pt	10pt
\small	9pt	10pt	11pt
\normalsize	10pt	11pt	12pt
\large	12pt	12pt	14pt
\Large	14pt	14pt	17pt
\LARGE	17pt	17pt	20pt
\huge	20pt	20pt	25pt
\Huge	25pt	25pt	25pt

要注意的是选择字体大小的命令也会改变段落的行距，不过这只有在换段标记与选择字体大小的命令同在一组时才会如此。因此，小组结束符 } 不应该出现的太早。比如下面两个例子中段落结束符 \par 的位置的不同就对行距产生了不同的影响。

```
{\large Sometimes the font size commands
also change the line spacing. Some
other text.\par}
```

Sometimes the font size commands also change the line spacing. Some other text.

```
{\large Sometimes the font size commands
also change the line spacing. Some
other text.}\par
```

Sometimes the font size commands also change the line spacing. Some other text.

在上面第一个例子中，因为换段命令 \par 与 \large 在同一组，所以行距也加大了。但第二个例子中因为 \par 在花括号之外，所以段落的行距并没有加大。

在同一段落中将某一行文字或者某几个词语的字体放大或缩小所产生的效果并不好。但还是会出现需要将一个段落或者多个段落的字体放大或缩小的情况。此时最好还是用字体尺寸命令的环境用法，也就是说可以将字体尺寸命令中的反斜杠去掉之后当作环境名来使用。比如命令 \large 对应着环境 large；命令 \Large 则对应着环境 Large。

下面是一个例子：

```
\begin{Large}
You might want to use the
environment syntax for local font
changing commands.
\end{Large}
```

You might want to use the environment syntax for local font changing commands.

3.7.5 数学模式中字体命令

文本中的字体命令并不能改变数学模式中的字体。对数学工作者来说，不同形状的字母常常有不同的含义。例如，直立的黑体字母常被用作向量的符号。如果因为周围条件的改变而使数学模式中的字体有所变化，其结果就有可能使人们不能正确领会数学公式的含义。因此，需要用不同的方法来改变数学模式中的字体。表 3.6 中列举了用在数学模式中的字体命令。注意命令 \mathcal 只对大写英文字母有效。另外，也不是所有数学符号都存在黑体形式。在 5.2.2 节中我们将介绍如何将一般的数学字符加黑。

表 3.6 数学模式中的字体命令

命　令	例　子	效　果
\mathcal{···}	$\mathcal{B}=c$	$\mathcal{B}=c$
\mathrm{···}	K_2	K_2
\mathbf{···}	$\sum x=\mathbf{v}$	$\sum x=\mathbf{v}$
\mathsf{···}	$\mathsf{G\times R}$	$\mathsf{G\times R}$
\mathtt{···}	$\mathtt{L}(b,c)$	$\mathtt{L}(b,c)$
\mathnormal{···}	$\mathnormal{R_{19}}\neq R_{19}$	$R_{19}\neq R_{19}$
\mathit{···}	$\mathit{ffi}\neq ffi$	$\mathit{ffi}\neq ffi$

还要注意所有声明形式的字体命令（如 \rmfamily ）都不能用在数学模式中。不过，带有一个参数的字体命令（如 \textrm{···}）则可以在数学模式中使用。使用这种命令可以暂时离开数学模式，像在文本中一样，输入一些逻辑上属于数学公式的文字。只是要注意所输入的文字实际上都带有数学模式之外的那些文字的属性。例如：

```
\sffamily The result will be
$$ x = 10 \textbf{ and thus }
y = 12 $$
```

The result will be

$$x = 10 \textbf{ and thus } y = 12$$

在上面这个例子中我们看到公式前面的 sans serif 族的属性传给了公式中的黑体字“and thus”。这使得“and thus”显得黑上加黑，从而与周围的数学公式很不协调。在这种情况下，也许还是 amstex 宏包提供的命令 \text{···} 使用起来效果更好些，它后面花括号中的文字与数学模式之外的文字的属性完全一样，看起来就很协调。

3.8 文稿的章节层次

为便于阅读和理解，多数书籍和文章的主体都被分成章、节和小节等。在主体的前面有标题页、前言和目录，而在主体的后面可能又有附录、索引和参考文献等。LaTeX 中有一些特殊的命令来排版章、节和小节这类标题，目录、索引和参考文献则可以自动生成。

3.8.1 标题页

标题页 (title page) 一般包含标题、作者、时间等， LaTeX 提供下述命令

```
\title{标题}   \author{作者和地址}   \date{时间}
```

来排版这些内容。标准的 LaTeX 都将这些内容分别排在一行的中间。如果标题太长，LaTeX 会自动将之分成若干行。用户也可以使用命令 \\ 指定在何处断行。当有两个以上的作者时，可以在作者之间插入命令 \and 将作者分开。例如下面的命令

```
\author{作者1\\ 所在单位\\ 地址 \and 作者2\\ 所在单位\\ 地址}
```

将第一个作者连同其单位和地址分三行排在左边，而第二个作者连同其单位和地址分三行排在右边。当然也可以使用命令 \\[长度] 将两个作者纵向排列。在 \date 之中可以输入时间和其他文字，但内容不可超过一行。如果不使用这条命令，系统会自动将当前的时间印出。如不想印出时间，只需使用 \date{} 就可以了。另外，还可以使用命令 \thank{脚注} 对标题、作者或时间加脚注。这条命令紧跟在欲加脚注的对象之后。在以上这些命令之后，还必须输入命令

```
\maketitle
```

才能最后产生标题页。例如：

```
\title{How to Write DVI Drivers}
\author{Helmut
Kopka\thank{Tel:05556-401-451}\\
Max--Planck--Institut\\
f\"ur Aeronomie
\and
Phillip G. Hardy\thank{Tel.
319--824--7134}\\
University\\
of Iowa}
\maketitle
```

How to Write DVI Drivers

Helmut Kopka[1]
Max-Planck-Institut
für Aeronomie

Phillip G. Hardy[2]
University
of Iowa

October 29, 1992

[1] Tel. 05556-401-451
[2] Tel. 319-824-7134

如果用户对用这种方式产生的标题页格式不满意，可以使用标题页环境

```
\begin{titlepage} 标题页内容 \end{titlepage}
```

来自己制作自由格式的标题页。

还需说明的是，对于 book 和 report 文件类型，标题页被单独地排印在最前面无页码的一页，而对于 article 文件类型，一般标题、作者和时间被排在第一页，并不单独分页，但如果文件类型 (\documentclass) 的选项中有选项 titlepage，则 article 文件类型也会有单独的标题页。

3.8.2 摘要

LaTeX 文稿的摘要 (abstract) 可以用环境

```
\begin{abstract} 摘要部分 \end{abstract}
```

排出。对于 report 文件类型，摘要一般被独立地排在标题页之后的一页上，并且没有页码。对于 article 文件类型，摘要通常被排在时间的下面居中的位置，不单独分页，除非文件类型选项中有选项 titlepage。book 文件类型中没有定义这个环境。摘要的标题是居中排版的黑体“Abstract”，可以用命令 \abstractname 重新设置它的形式。当然，不用 abstract 环境也很容易排版摘要。

3.8.3 章节

在 LaTeX 中，下述命令

```
\part   \chapter   \subsection      \paragraph
        \section   \subsubsection   \subparagraph
```

用来划分文件的部分、章、节和小节等。除了命令 \part 之外，上述命令形成文件的章节层次。对于 report 和 book 文件类型，最上面一层，是章 (\chapter)，章被分成节 (\section)，节又被分成小节 (\subsection)，等等。而 article 文件类型的最上面层次是节，其次是小节，等等。由于它没有章这个层次， article 文件类型的文件就很容易被编进书中作为一章。

上面这些命令都会产生序号，其格式是“*层次名 序号*”。除了最上面的层次外，层次的序号总是在上一层次的序号后加上一点和一个数。例如： Chapter 1，Section 1.1 等。这里 Chapter 和 Section 是层次名， 1 和 1.1 是序号。层次的序号可以通过下面的命令

```
\setcounter{层次名}{数}
```

进行重置。例如，在命令 \setcounter{chapter}{2} 之后章的序号将从 3 开始。

在 LaTeX 中，命令

```
\chaptername, \sectionname
```

指的就是层次名，而命令

```
\thechapter, \thesection
```

则指向层次的序号。最上面两层的层次名、序号以及层次的标题一般都会被自动写进目录和页眉中。如果层次的标题太长，就会产生溢出的水平盒子。因此对于很长的层次标题，最好按照下面命令

```
\sec_command[短标题]{标题}
```

附加一个短标题，这里 \sec_command 是层次命令。短标题被写进目录和页眉中，而正文主体中仍是原标题。

另外，上面这些层次命令都有带星号的形式，如 \chapter*、\section* 等。它们被用来排印不带序号的标题，这种标题不会被写进目录和页眉中。

\part 是一个特殊的层次命令，这条命令不影响其他层次的序号，通常只用于将长篇巨著分成几个部分。

层次命令本身从上至下也是有序的，\section 是第 1 层，\subsection 是第 2 层，……，\subparagraph 是第 5 层。对于 article 文件类型，\part 是第 0 层的，但对于 book 和 report 文件类型，\part 是第 −1 层的，而 \chapter 是第 0 层的，见表 3.7。层次深度 secnumdepth 就指向层次命令的这种序。在默认的情况下 book 和 report 文件类型的层次深度是 2，而 article 文件类型的层次深度是 3。不过，层次深度也可以用命令 \setcounter 来重新设置。层次深度以下的层次标题不被编序。

表 3.7 章节层次

层次	book, report	article	层次	book, report	article
-1	part		3	subsubsection	subsubsection
0	chapter	part	4	paragraph	paragraph
1	section	section	5	subparagraph	subparagraph
2	subsection	subsection			

3.8.4 附录

编排文件的附录可以用命令 \appendix。这是一个不带参数的命令，它只是重置了层次序号将之改为大写英文字母，并且层次名改为“Appendix”，如 Appendix A、Appendix B 等。附录下属层次的序号形如 A.1、B.2.1 等。编排附录也可以使用下面的附录环境

```
\begin{appendix} 附录内容 \end{appendix}
```

3.9 排 印 目 录

3.9.1 目录的自动生成

在 LaTeX 中，文件的目录 (table of contents) 可以由系统自动生成。默认的情况下，自动生成的目录中包含层次结构中最上面的三个层次的标题和相应的页码，也就是说对 book 和 report 文件类型，目录中包含与 \chapter、\section 和 \subsection 这些命令相对应的信息，而对于 article 文件类型，目录中包含与 \section、\subsection 和 \subsubsection 这些命令相对应的信息。用户可以通过下面的命令

```
\setcounter{tocdepth}{num}
```

来改变目录包含的层次，其中 *num* 是与层次相关的数。因此对于 book 和 report 文件类型来说， *num* 的默认值是 2；而对于 article 文件类型， *num* 的默认值是 3。

下面的命令

```
\tableofcontents
```

会在此命令出现的位置将目录排印出来。由于目录通常放在标题页和摘要之后，因此，当系统读到上面这条命令时，还不知各章节的内容，因而也无法立刻将目录编排好。事实上，第一次运行 LaTeX 程序时，系统只是建立了一个以主文件名为名以 toc 为扩展名的文件，并把各章节的标题信息存于此文件中，当第二次运行 LaTeX 程序时，系统读到命令 \tableofcontents 就将此文件中的各章节的标题信息取出排成目录。所以，要排印目录至少要运行两次 LaTeX 程序。

3.9.2 目录中的附加条目

在 3.8.3 节中，我们曾经提到 \chapter*、\section* 等这些带星号的层次命令会产生不被编号的标题，这些标题不能自动进入目录和页眉。如果想把这些没有编号的标题或其他的一些附加条目排印到目录中去，就需要用下面两条命令：

```
\addcontentsline{toc}{sec_name}{标题}
\addtocontents{toc}{标题}
```

上面第一条命令中的 *sec_name* 是层次名（不带反斜杠），如 chapter、section 等，这条命令将其中的标题连同页码作为完全与 *sec_name* 等同的层次写进目录中作为一个条目，只是没有序号而已。如果还想给这个条目附加一个序号，就必须用

```
\protect\numberline{序号}{标题}
```

来代替上面第一条命令中的标题。

上面方框中的第二条命令可以将任何需要的命令和文字写进 toc 文件中。写进去的内容可能是一些诸如 \protect\newpage 之类的格式化命令，当系统调用 toc 文件时，这些命令就被启用了。

3.9.3 表格和图形的目录

类似于文件目录的自动生成，LaTeX 系统用下述两条命令：

```
\listoftables 和 \listoffigures
```

来自动生成表格目录 (list of tables) 和图形目录 (list of figures) 。当 LaTeX 系统读到第一条命令时就根据表格环境中的命令 \caption 提供的信息建立或读取一个 lot 文件，而

当系统读到第二条命令时则根据图形环境中的 \caption 命令来建立或读取一个 lof 文件。系统程序运行两次后就在上面命令出现的位置排印表格或图形的目录。

类似于文件目录中附加条目，也可以根据下面更一般的命令

\addcontentsline{*file*}{*type*}{*text*} 和 \addtocontents{*file*}{*text*}

添加一些新的条目到表格或图形的目录中，其中 *file*，*type* 和 *text* 的含义如下：

file 目录文件的扩展符，通常是 toc，lof 和 lot 三者之一。

type 目录条目的格式。对于 toc 文件，*type* 指的是 chapter，section 等层次名。对于 lof 文件，*type* 指的是 figure。对于 lot 文件，*type* 指的是 table。

text 要输入到目录中的条目的内容。

3.10 交 叉 引 用

在某些书籍、报告或文章中作者常常会提醒读者参阅某页的图片、表格或特别的小节和段落等。这种在文本中引用另外一个地方的公式号、章节号、图表号等的引用称之为交叉引用 (Cross References)。由于事先并不知道要引用的部分在最后的排版中将会出现在哪一页，也由于作者在写作时经常会添加或删除某些段落及图表从而影响页码及序号的编排，所以在以前的一些排版系统中这种文本间的交叉引用是相当复杂的。如同目录的自动生成一样，LaTeX 系统有很好的方法来解决交叉引用的问题。下面的三条命令

\label{标签}, \ref{标签} 和 \pageref{标签}

就是用来在文稿中做交叉引用的，其中第一条命令紧跟在被引用的标题之后，为被引用的内容建立一个标签，在文稿中某处引用标题的序号时只需用第二条命令来引用已做好的标签。无论怎样修改文件，标签始终与序号保持一致。第三条命令用来引用标签所在的页码。这里的标签可以是区分大小写的字母、数字、除保留字符之外的符号，以及它们的组合。例如：

```
A reference to this subsection
\label{sec:this} looks like:
``see section~\ref{sec:this} on
page~\pageref{sec:this}.''
```

A reference to this subsection looks like: “see section 3.10 on page 64.”

当 \label 命令用于带标号的数学公式环境 (如 equation, eqnarray)、列表环境（如 enumerate）和图表环境时，被引用的分别是公式号、项目编号和图表号；当 \label 不在这些带标号的环境中时，被引用的是 \label 命令所在的章节号。

注意，不能使用两个同样的标签，否则，系统会提示出现了多重定义的标签。另外，也必须运行两次系统程序才能得到正确的引用标号。

3.11 排印参考文献

许多文章和书籍特别是科技方面的著作都包含大量参考文献。参考文献一般列在正文主体的后面。引用参考文献的方法有很多，不同的出版社对参考文献的格式也有不同的要求。通常引用参考文献时，都是在引用处标明参考文献的自然序号。当然可以用手工的方式来完成这种工作，即先将所有参考文献排序，然后在正文中的引用处输入参考文献的序号。但如果参考文献很多，而且在修改时又经常增加或删除某些参考文献，那么手工的方式就很烦琐而且也很容易出错了。在 LaTeX 中对参考文献的引用实际上也是一种交叉引用，因而也非常方便。首先将参考文献列在 thebibliograph 环境中，用法如下：

```
\begin{thebibliography}{widest_entry}
参考文献列表
\end{thebibliography}
```

其中 *widest_entry* 是最长的参考文献编号，一般取 9、99、999 等数字。每一个参考文献可以通过下述方式输入：

```
\bibitem[名称]{标签} 参考文献
```

这里标签参数的作用与 \label 命令中的标签参数作用是一样的，它可以是区分大小写的字母、数字以及除保留字符和逗号之外的符号，以及它们的组合。*名称*参数是可选项，其作用就是给参考文献起一个名。如果不给出*名称* 参数，LaTeX 就以参考文献在列表中的自然序号（阿拉伯数字）作为名称。在 book 和 repart 文件类型中，参考文献部分是作为以 Bibliography 为标题的一章；而在 article 文件类型中，参考文献部分则是作为以 References 为标题的的一节。例如在 article 文件类型中输入下面的命令：

```
\begin{thebibliography}{9}
\bibitem{lamport} Leslie Lamport, \textit{A Document Preparation
        System}, Addison-Wesley Co., Inc., Reading, MA, 1985.
\bibitem{knuth} Donald E. Knuth, \textit{Computers and Typesetting},
      Vol. A-E. Addison-Wesley Co., Inc., Reading, MA, 1984--1986.
\bibitem[2a]{knuth:a} Vol A: \textit{\TeX{}book}, 1984.
\bibitem[2b]{knuth:b} Vol B: \textit{\TeX{}: The Program}, 1986.
\end{thebibliography}
```

将排印出：

References

[1] Leslie Lamport, *A Document Preparation System*, Addison-Wesley Co., Inc., Reading, MA, 1985.

[2] Donald E. Knuth, *Computers and Typesetting*, Vol. A-E. Addison-Wesley Co., Inc., Reading, MA, 1984–1986.

[2a] Vol A: *TEXbook*, 1984.

[2b] Vol B: *TEX: The Program*, 1986.

在正文中引用参考文献时，可以用下述命令：

```
\cite[附加信息]{标签1,标签2,···}
```

其中附加信息是可选项，它给出除了参考文献名称以外的信息。以上面阴影中的参考文献为例，在正文中

输入: `See \cite{lamport,knuth}` 得到: See [1, 2]

输入: `See \cite{knuth:a}` 得到: See [2a]

输入: `See \cite[page 3--4]{knuth}` 得到: See [2, page 3–4]

3.12 索引的生成

在阅读长篇著作特别是科技文献时，我们经常需要查阅目录和索引。因此，索引也是文献的一个重要组成部分，其中通常列出了经常出现的名词和重要概念及其在文献中的对应页码。通过查阅索引读者很容易查到所关心的部分，从而提高阅读效率。在编写 LaTeX 文件时借助于程序 makeindex 我们可以很容易排印出各种式样的索引。这里我们只阐述在 LaTeX 文件中排印索引的基本方法。关于更详尽的说明，读者可参阅其他文献，如 The LaTeX Companion[7]。

欲排印索引，首先要在 LaTeX 源文件的导言部分中使用命令

```
\usepackage{makeidx}
```

来加载 makeidx 宏包，并且还要使用命令

```
\makeindex
```

来通知 LaTeX 提取源文件中的索引条目。如果要使索引的一个条目指向文中的某个词，需在这个词的后面使用命令

```
\index{条目内容}
```

它可以将条目内容以及所在页的页码排印到索引中。条目内容呈现的形式可以是多样的，比如，`\index{package!amsmath}` 将 amsmath 作为 package 的一个子条目，并且子条目向里有一个缩格。表 3.8 给出了几种索引条目样式。

表 3.8 索引实例

例　　子	条目内容	说　　明
`\index{package}`	package, 1	一般形式
`\index{package!amsmath}`	amsmath, 3	“package” 的子条目
`\index{Sam@\textsl{Sam}}`	*Sam*, 2	斜体条目
`\index{Lin@\textbf{Lin}}`	**Lin**, 7	黑体条目
`\index{Jenny|textbf}`	Jenny, **3**	黑体页码
`\index{Bezier@B\'ezier}`	Bézier, 8	重音词条目

当第一次运行 LaTeX 程序时，每个条目以及所在页码信息都被写进自动创建的文件 `filename.idx` 中，其中 *filename* 就是源文件名。然后运行外部程序 makeindex，即在命令行中键入命令

```
makeindex filename
```

对文件 `filename.idx` 进行排序等处理，并产生一个新的文件 `filename.ind`。第二次运行 LaTeX 程序时，LaTeX 就将这个新产生的索引文件中的内容排印在源文件中命令

```
\printindex
```

所在之处。

3.13 脚注和旁注

3.13.1 脚注

脚注可以用命令

```
\footnote{脚注内容}
```

来排印。这条命令必须紧跟在要被注解的文字或标点符号之后。LaTeX 会在被注解的文字的右上角印上小号的脚注标签，同时在当前页的左下角排印出相应的标签及脚注。

脚注与主体文字被一条称为脚注线的较短的实心横线分开。脚注内容可以有多段文字，并且可以包含列表、数学公式和表格等。对于 article 类型的文件，脚注被从头到尾顺序编号；但对于 book 和 report 文件，脚注是在每一章中各自顺序编号的。如果要使脚注在每一页中各自顺序编号，可以使用 footnpag 宏包。

命令 \footnote 只能在段落模式中使用，而不能在左右模式和数学模式中使用，也不能在脚注中又使用脚注。不过它可被用在小页环境中，此时脚注被排印在小页的左下角，而不是整页的左下角。命令 \thefootnote 指向脚注标签。通常脚注标签都是阿拉伯数字，但可以用命令

```
\renewcommand{\thefootnote}{\number_style{footnote}}
```

来重新定义标签的形式，其中 \number_style 表示命令 \arabic、\roman、\Roman、\alph、\Alph 和 \fnsymbol 之一。 \fnsymbol 是一个比较特殊的命令，它指示用下面 9 个符号

* † ‡ § ¶ ‖ ** †† ‡‡

来代替第 1 到第 9 个标签，因此有必要在第 10 个标签出现之前重新设置标签。

在小页环境中指向标签的命令是 \thempfootnote。默认情况下小页环境中的标签是顺序编号的小写字母，并且独立于小页环境之外的脚注编号。不过我们可以用命令

\footnotemark 和 \footnotetext

来使小页环境中的脚注和小页环境之外的脚注统一起来。命令 \footnotemark 在小页环境中产生一个脚注标签，其形式与序号都与环境之外的脚注标签同等对待，脚注的内容要用命令 \footnotetext 来给出。在下面的例子中，我们对小页环境中的两个词进行注解，一个注解放在小页环境的左下脚，注解标签是小写字母 a，另一个注解放在整页的左下脚，注解标签是数字 1。

```
\begin{minipage}{\linewidth}
Footnotes in a minipage are
numbered using lowercase
letters.\footnote{Inside minipage}
\par This text references a
footnote at the bottom of
the page.\footnotemark
\end{minipage}\footnotetext{At
bottom of page}
```

Footnotes in a minipage are numbered using lowercase letters.[a]

This text references a footnote at the bottom of the page.[1]

[a]Inside minipage

注意，\footnotemark 用于小页环境中而 \footnotetext 用于环境之后。

对于附加的脚注可以使用带选项的命令

\footnote[*num*]{脚注内容}

来排印，其中 *num* 是正整数，表示指定的标签序号，这条命令不影响其他的脚注序号，例如：

```
\renewcommand{\thefootnote}%
{\fnsymbol{footnote}} An optional
argument\footnote[8]{The 8th
symbol} may be added to the
footnote command.
```

An optional argument[††] may be added to the footnote command.

[††]The 8th symbol

脚注的样式可以通过下面的一些命令参数来改变：

\footnotesize　用来定义脚注字体的大小。

\footnotesep　用来定义脚注之间的垂直距离。

\skip\footins　这是一个底层 TeX 命令，用来定义主体文字与第一条脚注之间的距离。可以用 \setlength 或 \addtolength 这两个命令来改变它的值。

\footnoterule　这是画脚注线命令，用来分割主体文字与脚注，通常这条命令在 \skip\footins 之后立即被执行，而且它本身不应当占有垂直空白，因而可以用一个负距离来抵消它所占的垂直空白。

例如用下面的命令

```
\renewcommand{\footnoterule}{\vspace*{-3pt}%
\rule{.4\columnwidth}{0.4pt}\vspace*{2.6pt}}
```

来重新定义脚注线。又例如下面的命令

```
\renewcommand{\footnoterule}{\vspace*{-3pt}%
\dotfill\vspace*{2.6pt}}
```

定义脚注线为与页面宽度同长的点状虚线。

3.13.2 旁注

旁注就是写在页面边缘空白处的注解或说明。它可以通过命令

\marginpar[左旁注内容]{右旁注内容}

[1]At bottom of page

来排印。这条命令将旁注排印在左边空白处或右边空白处，旁注的第一行与命令所在行对齐。默认情况下，如果命令中只包含*右旁注内容*，则单面排版时旁注被排印在右边，双面排版时旁注被排印在页面的外侧。所谓外侧是指奇数页的右边和偶数页的左边。对于双栏排版的页面，旁注被排印在左栏的左边和右栏的右边。如果命令中既有*左旁注内容* 又有*右旁注内容*，则*左旁注内容*被排印在页面的左边，而*右旁注内容*被排印在页面的右边。例如用下面的命令

```
\marginpar[\raggedleft$\Longrightarrow$]{$\Longleftarrow$}
```

会在旁边排印一个箭头，不管它出现在奇数页还是偶数页箭头始终指向文字。

有三个问题需要说明：首先，如果命令 \marginpar 出现在段落的最前面，即段落的第一行第一个字的前面，那么旁注的第一行也许不能与命令所在行对齐。第二，由于旁注所在的空白太窄，而某些单词太长，用户可能需要在旁注的第一个词之前插入命令 \hspace{0pt} 从而使 LaTeX 能拼读第一个词。第三，出现在左边的旁注可能不是右对齐的从而影响美观。对于第一个问题，可以在命令 \marginpar 前面放置一个空盒子 \mbox{} 来解决。对于第三个问题，可以在旁注内容的前面使用命令 \raggedleft 解决。也可以按照如下方法

```
\newcommand{\marginlabel}[1]%
{\mbox{}\marginpar[\raggedleft\hspace{0pt}#1]{\hspace{0pt}#1}}
```

定义一个新命令

\marginlabel{*旁注内容*}

将这三个问题一并解决。它在原先的命令 \marginpar 之前插入一个空盒子 \mbox{}，而在*左旁注内容*之前插入命令 \raggedleft\hspace{0pt}，并在*右旁注内容*之前插入命令 \hspace{0pt}。

如前所述对于单页排版的页面，在默认情况下旁注被排印在右边空白处。这种默认设置可以被下面的命令改变：

\reversemarginpar　将旁注排印在与默认方向相反的页边空白处。

\normalmarginpar　按默认方向来排版旁注。

下面是另外三个有关旁注的参数，可以用 \setlength 命令来设置它们的值：

\marginparwidth　用来定义旁注的宽度。这个参数的值不能设置的太大，否则部分旁注会超出页面边界，而不能排印出来。

\marginparsep　用来定义旁注与主体文字之间的距离。这个参数的值设置的太大，会使旁注的空间不够，但设置的太小却会使旁注与主体文字拥挤在一起。

\marginparpush　用来定义旁注与旁注之间的最小垂直距离。

3.14 显 示 文 本

3.14.1 左对齐、右对齐和居中环境

在排版时，为了某种特殊的原因可能希望一个段落的文字左边对齐而允许右边不对齐，也可能希望一个段落右边对齐而允许左边不对齐。下面的 flushleft 环境和 flushright 环境

```
\begin{flushleft} 第 1 行\\ 第 2 行\\ ··· \end{flushleft}
\begin{flushright} 第 1 行\\ 第 2 行\\ ··· \end{flushright}
```

就分别将环境中的每行文字按左对齐和右对齐进行排版。如果一行中的文字太长，LaTeX 自动将其分成多行，且不对一行中的最后一个词进行拼读分割。例如：

```
\begin{flushleft}
The environments flushleft and\\
flushright generate paragraphs\\
which are either left-aligned or
right-aligned. If the text is
too long for one line, it is split
over several lines.
\end{flushleft}
```

The environments flushleft and
flushright generate paragraphs
which are either left-aligned or right-aligned.
If the text is too long for one line, it is split
over several lines.

```
\begin{flushright}
The environments flushleft and\\
flushright generate paragraphs\\
which are either left-aligned or
right-aligned.
\end{flushright}
```

The environments flushleft and
flushright generate paragraphs
which are either left-aligned or right-aligned.

上述两个环境可以分别用下面两条命令取代：

```
{\raggedright 第 1 行\\ 第 2 行\\ ··· \\}
{\raggedleft 第 1 行\\ 第 2 行\\ ··· \\}
```

注意最后的换行符 \\ 不能少。

此外下面的 center 环境

```
\begin{center} 第 1 行\\ 第 2 行\\ ··· \end{center}
```

可以将几行文字居中排印。与 \flushleft 一样，如果一行中的文字太长，LaTeX 自动将其分成多行。这个环境也可用命令

```
{\centering 第 1 行\\ 第 2 行\\ ··· \\}
```

取代，注意最后的换行符 \\ 也不能少。要使单独一行居中也可用命令

```
\centerline{居中文字}
```

无论是左对齐、右对齐还是居中，如果要在行与行之间增加一个附加距离，例如增加 2 毫米，则可在换行符后面紧跟一个 [2mm] ，即 \\[2mm]。

3.14.2 缩格环境和诗歌环境

要使一段或若干段文字在排版时两边都向里缩进相同的距离，可以用下面的 quote 环境和 quotation 环境：

```
\begin{quote}  被缩格排版的文字  \end{quote}
\begin{quotation}  被缩格排版的文字  \end{quotation}
```

用这两个环境排版时，被缩格排版的文字与上下文之间的距离比正常的行距要大些。这两个环境的不同之处在于 quotation 环境中段落的第 1 行的左边像通常文字一样又向里缩格排印，而 quote 环境中的第 1 行就不再缩格了：

```
In the quatation environment,
paragraphs \begin{quote} are maked
by extra indentation of the first
line,\end{quote} \begin{quotation}
whereas in quote environment,
they are indicated with more
vertical spacing between them.
\end{quotation}
```

In the quatation environment, paragraphs are maked by extra indentation of the first line,

whereas in quote environment, they are indicated with more vertical spacing between them.

另外一个缩格排版的环境是 verse 环境，它一般用来排版诗歌：

```
\begin{verse} 第 1 行\\ 第 2 行\\ ··· \end{verse}
```

在此环境中，段落文字只在左边被缩格排版而右边仍按通常段落文字来排并不缩格。如果一行的文字太长，LaTeX 会在适当处（也可能在一个词的中间）将句子分开，排不下的文字放到下一行进一步缩格排版。例如：

```
\begin{verse}
I am the wind that wavers,
You are the certain land;\\
I am the shadow that passes
Over the sand.\\
\end{verse}
```

I am the wind that wavers, You
are the certain land;
I am the shadow that passes Over
the sand.

缩格环境可以嵌套使用，也就是说一个缩格环境中可以使用另一个缩格环境，但最多只能嵌套 6 次。

3.14.3 定理型表述环境

在科技文献中常出现一些定理型段落结构，用来表述定理、公理、定义等。如：

定义 3.1 设 f 是定义在度量空间 X 上的实函数，p 是 X 中一点。如果存在一个正数 $\delta > 0$ 使得对一切满足 $d(p,q) < \delta$ 的点 $p \in X$，有 $f(q) \leqslant f(p)$，则称 p 是 f 的一个极值点。

又如：

Theorem 1 (Lebesgue). *Suppose that f is a bounded real function defined on the interval $[a,b]$. Then f is integrable over $[a,b]$ in the sense of Riemann if and only if the Lebesgue measure of the set of discontinuous points of f in $[a,b]$ is zero.*

这些定理型表述的特点是：它们是独立的段落，有一个名称（如 Theorem、定义），有序号，段落内容可以有与正文不同的字体，名称与内容之间以及定理型表述与上下文之间都有合适的距离。当然所有这一切都可以在正文中逐个手工调整。但如果因为某种原因需要在前面插入一个定理时，那么它后面的那些定理型表述的序号可能都要手工改变，而且手工输入的序号也使交叉引用带来不便。

事实上，LaTeX 提供了一种输入定理型表述的方法使我们能避免一些不必要的麻烦。下面这条命令

```
\newtheorem{env_name}{名称 1}[in_counter]
```

通常置于源文件的导言部分，用来定义一个定理型表述环境。其中 *env_name* 是环境名，*名称 1* 是定理型表述的名称，可选项 *in_counter* 是章节计数器（一般是 section 和 chapter）。比如，此计数器是 section 时表明这个定理型表述的序号只在节（section）中顺序变化，每一个 \section 命令都使以*名称 1* 为名的定理型表述的序号重新从 1 开始。计数器省略时定理型表述的序号在整个文稿中按顺序编号。一旦上面这个定理型表述环境定义好了，我们就可以用

```
\begin{env_name}[附加标题]  表述的内容  \end{env_name}
```

来输入定理型表述。另外，下面的命令

```
\newtheorem{env2_name}[env_name]{名称 2}
```

定义一个环境名为 *env2_name* 名称为*名称 2* 的定理型表述环境，而且这种定理型表述与环境 *env_name* 中的定理型表述是混合编号的。

通常定理型表述是一个不缩格的段落，其名称使用黑体字，表述内容使用意大利式斜体字，而且序号后面有一个黑点将标题与表述内容分开。但这种定理型表述的格式是可以改变的，如前面的定义 3.1 和 Theorem 1 就有不同的格式。Tools 宏包套件中的 theorem 宏包以及另一个功能更强大的 ntheorem 宏包都定义了一些不同的定理格式。在 150 页 5.5 节中我们将介绍如何用 AMSLaTeX 中的 amsthm 宏包来定义不同的定理型表述格式。

3.14.4 强调文本

通常 LaTeX 系统使用带参数的命令 \emph 或它的声明形式 \em（见 3.7.3 节）来强调一段文字，而不建议直接使用改变字型的命令如 \bfseries、\itshape 等。如果要用下划线来强调一段文字，可以使用命令

```
\underline{加下划线的文字}
```

另外，也可以使用 ulem 宏包。此宏包重新定义了命令 \emph，使得我们能用各种下划线来强调文字。由于被强调的文字实际上是置于一个带下划线的左右盒子中，TeX 不能自动对词语进行拼读分割，但我们可以在词语中适当处插入 \- 来通知 TeX 此处可以分割。另外，在被强调的部分允许出现空格和弹性空白长度，也可以使用 \newline、\linebreak 等换行命令，但是不能包含数学模式也不能换段。如果对被强调部分中的某些语句再进行强调，那么命令 \emph 应该作用于第二层强调语句中的每一个词。例如：

```
\emph{In the nested emphasis,
\texttt{\textbackslash emph} had
to be given \emph{for} \emph{each}
\emph{word} separately so the spaces
between could stretch and break
into lines.}
```

In the nested emphasis, \emph had to be given for each word separately so the spaces between could stretch and break into lines.

下划线的粗细是由 \ULthickness 定义的，可以用 \renewcommand（不是用 \setlength）来改变，例如 \renewcommand{\ULthickness}{2pt} 将下划线的粗细设置成 2pt。被强调文字到下划线之间的垂直距离由参数命令 \ULdepth 控制，比如

命令 \setlength{\ULdepth}{1pt} 将被强调文字到下划线之间的距离设置为 1pt。如果某些语句不想用下划线强调，而想用以前的方式，那么在强调语句之前加上命令 \normalem，它把命令 \emph 的定义切换成以前的。反之，可以用命令 \ULforem 再将强调切换成下划线的情形。

ulem 宏包还定义了几种其他形式的下划线命令：

\uline　在文本下面画一条水平直线。
\uuline　在文本下面画两条水平直线。
\uwave　在文本下面画一条波浪线。
\sout　画一条水平直线穿过文本。
\xout　用斜杠 //// 在文本上做标记，示意将文本删除。

例如：

```
\uline{underlined text}\\
\uuline{double underlined text}\\
\uwave{wavy underlined text}\\
\sout{line through words}\\
\xout{text marked to remove}
```

underlined text
double underlined text
wavy underlined text
~~line through words~~
~~text marked to remove~~

注意，当使用 \xout 命令时，其中的文字已很难辨认，这通常是在文字上做记号示意将文字删除，并且不影响排版效果，必要时再将命令取消，从而恢复原来的文字。

3.14.5 特殊形状的段落

将一段文字按照特别的形状来排版，这对于大部分的排版系统来说都不是很容易的。shapepar 宏包定义了一条命令 \shapepar 使得我们能在 LaTeX 中很容易就把一段文字排列成特殊形状。

命令 \shapepar 必须用在一段文字的开始，其语法如下：

```
\shapepar{形状参数} 段落文本
```

其中形状参数是对其后的段落文本的一种形状描述。形状参数本身的定义必须遵从一定的语法规则。当然要定义一个新的形状参数也不是很容易的，shapepar 宏包中已定义了几种形状参数，这几种形状参数命令是：\diamondshape（钻石形）、\heartshape（心形)、\circleshape（圆形）、\starshape（五角星形）、\nutshape（坚果形）和 \squarshape（正方形），还定义了几个相对应的简单的形状命令 \diamondpar、\heartpar、\circlepar、\starpar、\nutpar 和 \squarepar。下面是一个段落被分别排成不同形状的例子：

君不见黄河之水天上来，奔流到海不复回。君不见高堂明镜悲白发，朝如青丝暮成雪。人生得意须尽欢，莫使金樽空对月。天生我材必有用，千金散尽还复来。烹羊宰牛且为乐，会须一饮三百杯。岑夫子、丹丘生。将进酒，君莫停。与君歌一曲，请君为我侧耳听。钟鼓馔玉不足贵，但愿长醉不愿醒。古来圣贤皆寂寞，惟有饮者留其名。陈王昔时宴平乐，斗酒十千恣欢谑。主人何为言少钱，径须沽取对君酌。五花马，千金裘，呼儿将出换美酒，与尔同销万古愁

\squarepar{君不见黄河……}

君不见黄河之水天上来，奔
流到海不复回。君不见高堂明
镜悲白发，朝如青丝暮成雪。人生
得意须尽欢，莫使金樽空对月。天生
我材必有用，千　　金散尽还复来。
烹羊宰牛且为　　乐，会须一饮
三百杯。岑夫　　子、丹丘生。
将进酒，君莫　　停。与君歌一
曲，请君为我　　侧耳听。钟鼓
馔玉不足贵，　　但愿长醉不愿
醒。古来圣贤皆　　寂寞，惟有饮者
留其名。陈王昔时宴平乐，斗酒十千恣
欢谑。主人何为言少钱，径须沽取对
君酌。五花马，千金裘，呼儿将
出换美酒，与尔同销万古愁

\shapepar\nutshape{君不见黄河……}

君不见黄　　河之水天
上来，奔流到海不　　复回。君不见高堂
明镜悲白发，朝如青丝暮成雪。人生得意须尽
欢，莫使金樽空对月。天生我材必有用，千金散
尽还复来。烹羊宰牛且为乐，会须一饮三百杯。
岑夫子、丹丘生。将进酒，君莫停。与君歌一
曲，请君为我侧耳听。钟鼓馔玉不足贵，但愿
长醉不愿醒。古来圣贤皆寂寞，惟有饮者留
其名。陈王昔时宴平乐，斗酒十千恣欢
谑。主人何为言少钱，径须沽取对
君酌。五花马，千金裘，呼
儿将出换美酒，与尔
同销万古愁
♡

\heartpar{君不见黄河……}

◇
君　不
见黄河之水
天上来，奔流到
海不复回。君不见高
堂明镜悲白发，朝如青丝
暮成雪。人生得意须尽欢，莫使
金樽空对月。天生我材必有用，千金
散尽还复来。烹羊宰牛且为乐，会须一饮
三百杯。岑夫子、丹丘生。将进酒，君莫停。
与君歌一曲，请君为我侧耳听。钟鼓馔玉不足
贵，但愿长醉不愿醒。古来圣贤皆寂寞，
惟有饮者留其名。陈王昔时宴平乐，
斗酒十千恣欢谑。主人何为言
少钱，径须沽取对君酌。
五花马，千金裘，呼
儿将出换美酒，
与尔同销万
古　愁
◇

\diamondpar{君不见黄河……}

注意，在这些特殊形状的段落中不能使用数学模式以及垂直摆放的内容，如 \vspace 和美元符号等命令。

3.14.6 文本的垂直叠放

标准的 LaTeX 提供了一个将多行文本垂直叠放的命令：

```
\shortstack[位置]{文本行\\ 文本行\\ ···}
```

其中位置是可选项，默认值是 c，表示叠放的文本行居中对齐，它也可以取选项 l 和 r，分别表示叠放的文本行左对齐和右对齐。例如：

```
\shortstack[l]{top\\center line\\
bottom}\hspace{\fill}
\shortstack{top\\center line\\
bottom}\hspace{\fill}
\shortstack[r]{top\\center line\\
bottom}
```

top　　top　　top
center line　　center line　　center line
bottom　　bottom　　bottom

命令 \shortstack 实际上相当于只有一列的 tabular 环境（见 4.2 节），它常被用来将字母一个一个堆积起来，从而得到竖排的词，另外它也常被用来排旁注。

3.14.7 图文框

由 Friedhelm Sowa 编写的 picinpar 宏包定义了几种环境使我们能在一段文本中插入一个图文框。这个宏包定义的基本环境是 window 环境，它有两个变形，即 figwindow 环境和 tabwindow 环境。它们的语法如下：

```
\begin{window}[行数,位置,内容,描述]  段落文字  \end{window}
\begin{figwindow}[行数,位置,内容,描述]  段落文字  \end{window}
\begin{tabwindow}[行数,位置,内容,描述]  段落文字  \end{window}
```

其中行数是指在包含图文框的段落中，图文框上面的文本的行数，不过根据排版的需要，即使行数是正数，有时图文框仍会排在段落的顶部。位置这个选项表示图文框在段落中所居的位置，它的默认值是字母 l，这表示图文框将被排在段落的左边。另外两个选项是 c 和 r 分别表示图文框将被排在段落的中间和右边。内容是图文框中显示的内容。描述是对图文框内容的描述，一般是图文框的标题。段落文字 就是包含图文框的段落，它可以是一行文字，也可以是多行文字。当图文框中第一段文字结束时，如果图文框还没有排完，那么下一段文字就会在上段文字结尾的那一行开始排。

window 环境定义的图文框中可以包含文字，也可以包含表格和图片，但是用描述选项生成的标题不会进入表格目录和图形目录。在下面的例子中我们使用命令 \shortstack 把“good”垂直书写作为图文框的内容，然后将图文框嵌入一个段落的中间，且图文框上面有一行文字：

```
\begin{window}[1,c,{\fbox{%
\shortstack{g\\o\\o\\d}}},{}] In this case we
center a word printed vertically inside the
paragraph.  It is not difficult to understand
that tables can also be easily included with the
\texttt{tabwindow} environment.\par When a
paragraph ends, like here, and the window is not
yet finished, then it just continues past the
paragraph boundary, right into the next one(s).
\end{window}
```

In this case we center a word printed vertically inside the paragraph. It is not g o o d difficult to understand that tables can also be easily included with the tabwindow environment. When a paragraph ends, like here, and the window is not yet finished, then it just continues past the paragraph boundary, right into the next one(s).

下面我们用 figwindow 环境在一个文本段落中生成一个图文框，它的标题与其他的图形标题是统一的，因而会进入图形目录中。然后用 graphicx 宏包中 \includegraphics 命令将一个名为 ust.ps 的图片装入这个图文框：

```
\begin{figwindow}[1,r,{\includegraphics%
[width=3.0cm]{ust.eps}},{The Piazza}] On 23
September, the University launched its 10th
anniversary celebrations with ceremonies both on
and off campus. The entire university community
was immersed in a jubilant mood that brightened
up even the overcast sky. At 12:30pm, more than
1,000 students, faculty and staff gathered in
the Piazza for the Kickoff Celebration. The
program began with performances by the Students'
Union Band Society, the Wind Ensemble and the
String Team.
\end{figwindow}
```

On 23 September, the University launched its 10th anniversary celebrations with ceremonies both on and off campus. The entire university community was immersed in a jubilant mood that brightened up even the overcast sky. At 12:30pm, more than 1,000 students, faculty and staff gathered in the Piazza for the Kickoff Celebration. The program began with performances by the Students' Union Band Society, the Wind Ensemble and the String Team.

图 3.2 The Piazza

3.15 列　　表

列表是文本的一种重要结构。在生活中列表的使用也是非常普遍的。饭店的菜谱、商场的商品目录、公司的产品介绍等，无一不是借助列表来获得清晰、规范的表达。LaTeX 提供了很强的功能来排版列表结构。这一节我们要介绍 LaTeX 的三种列表环境：enumerate 环境、itemize 环境和 description 环境，以及怎样修改这些环境和按照自己的要求定义新的列表环境。

3.15.1 编号列表（enumerate 环境）

编号列表也称有序列表，它的每一列表项之前都被冠以一个编号，称为标号或前导符号。在 LaTeX 中，下面的 enumerate 环境

```
\begin{enumerate} 列表项目 \end{enumerate}
```

可以用来创建一个编号列表，其中列表项目中的每一项都必须用命令 \item 来引导，比如，“\item 这是第一项内容”。项目中的文字如果超过一行，那么换行后的第一个字符会与前行的第一个字符对齐，而让编号突出。编号列表以及后面要介绍的其他列表都可以嵌套使用。在默认情况下编号列表的第 1 层（即最外层）用带点的阿拉伯数字编号，第 2 层用圆括号中小写英文字母编号，第 3 层用带点的小写罗马字母编号，第 4 层用带点的大写英文字母编号。但这些编号形式都是可以改变的。下面是一个编号列表的例子：

```
\begin{enumerate}
\item The first item of the list.
This is a long item.
\item The second item of the list.
\item The third item of the list.
\end{enumerate}
```

1. The first item of the list. This is a long item.
2. The second item of the list.
3. The third item of the list.

完整的编号通常包含两个部分，第一部分就是编号，如数字或字母，它随着项目变化，另外一部分就是标号区，如黑点和括号等，它不随项目变化。表 3.9 中列出了控制编号列表环境的命令。其中，

计数器　用来对编号进行计数。
编号命令　指向编号的命令。
编号定义　是对编号命令的默认定义。
标号区　用来决定编号形式是否附带点、括号等的命令。
标号定义　是对标号区的默认定义。
编号例子　显示了两个编号形式的例子。
前置符　用来定义引用中出现的编号的前一个字符的命令。
前置定义　是对前置符的默认定义。
引用例子　引用中出现的编号表示的例子。

表 3.9　控制编号列表环境的命令

	第 1 层	第 2 层	第 3 层	第 4 层
计数器	enumi	enumii	enumiii	enumiv
编号命令	\theenumi	\theenumii	\theenumiii	\theenumiv
编号定义	\arabic{enumi}	\alph{enumii}	\roman{enumiii}	\Alph{enumiv}
标号区	\labelenumi	\labelenumii	\labelenumiii	\labelenumiv
标号定义	\theenumi.	(\theenumii)	\theenumiii.	\theenumiv.
编号例子	1., 2.	(a), (b)	i., ii.	A., B.
前置符	\p@enumi	\p@enumii	\p@enumiii	\p@enumiv
前置定义	{}	\theenumi	\theenumi(\theenumii)	\p@enumiii\theenumiii
引用例子	1, 2	1a, 2b	1(a)i, 2(b)ii	1(a)iA, 2(b)iiB

通过对这些命令的重新定义，可以得到各种各样的编号形式和它们的引用形式。在下面这个例子中我们将第 1 层编号形式改为大写罗马字母，将第 2 层的编号形式改为带点的大写英文字母，并将第 2 层的前置符改为第 1 层编号加一个短横线：

```
\makeatletter
\renewcommand{\theenumi}{\Roman{enumi}}
\renewcommand{\theenumii}{\Alph{enumii}}
\renewcommand{\labelenumii}{\theenumii.}
\renewcommand{\p@enumii}{\theenumi--}
\makeatother
\begin{enumerate}
\item\textbf{First level No.1}
\begin{enumerate}
\item Seconad level No.1 \label{q1}
\item Seconad level No.2 \label{q2}
\end{enumerate}
\item\textbf{First level No.2}
\label{q3}
\end{enumerate}
\ref{q1}\quad \ref{q2} \quad \ref{q3}
```

I. **First level No.1**

A. Seconad level No.1

B. Seconad level No.2

II. **First level No.2**

I–A I–B II

注意，在上面这个例子中我们使用了命令 \makeatletter 和 \makeatother 。这是由于我们要修改含有字符 @ 的命令 \p@enumii 。LaTeX 将字符 @ 放在 other 这一类，不作为普通字母来对待。通常含有字符 @ 的命令只用于宏包之中，若要在源文件中使用这种命令，必须把它们夹在命令 \makeatletter 和 \makeatother 之中。命令 \makeatletter 将其后面的 @ 转为普通字母，而命令 \makeatother 又将它还原为 other 类的字符。

下面这个例子中我们将第 1 层编号的前面加上符号 § 而在后面加上一个点，并将第 2 层编号改为圆括号括起来的数字。

```
\renewcommand{\labelenumi}{\S\theenumi.}
\renewcommand{\theenumii}{\arabic{enumii}}
\begin{enumerate}
\item First level item.
\begin{enumerate}
\item The first item in level 2.
\begin{enumerate}
\item The first item in level 3.
\end{enumerate}
\item The second item in level 2.
\end{enumerate}
\item Another first level item.
\end{enumerate}
```

§1. First level item.

(1) The first item in level 2.

i. The first item in level 3.

(2) The second item in level 2.

§2. Another first level item.

利用宏包 pifont 和 calc 还可以将编号形式定义成带圈的阿拉伯数字。例如：

```
\newcounter{local}
\renewcommand{\labelenumi}%
{\setcounter{local}{171+\value{enumi}}%
\ding{\value{local}}}
\begin{enumerate}
\item First level item.
\item Another first level item.
\end{enumerate}
```

➀ First level item.
➁ Another first level item.

此时编号的项数不能超过 10 个，这是因为宏包 pifont 中只提供了 10 个同类型的带圈阿拉伯数字。

总之，通过对控制编号列表环境的命令的修改几乎可以得到任何式样的编号形式。但是如果用户不想对这些命令本身做修改而又想得到有些变化的编号形式，那么可以使用 enumerate 宏包。这个宏包扩展了 LaTeX 中的 enumerate 环境，使之带有一个可选参数，来决定编号的形式。此时 enumerate 环境的形式是

```
\begin{enumerate}[编号形式]  列表项目  \end{enumerate}
```

其中可选参数编号形式可以取 A, a, I, i 和 1，分别表示编号形式为大写英文字母、小写英文字母、大写罗马字母、小写罗马字母和阿拉伯数字。当然编号形式也可以是其他的表达式，但是如果编号形式中用到了 A, a, I, i 和 1 这几个字符而又不将其作为编号对待，那么这几个字符必须用一对花括号 { } 括起来。例如环境

```
\begin{enumerate}[EX i.]  列表项目  \end{enumerate}
```

中的编号形式为 EX i., EX ii. 等，而环境

```
\begin{enumerate}[{A}-1]  列表项目  \end{enumerate}
```

中的编号形式为 A-1, A-2 等，其中的字母 A 已经不作为编号对待。

3.15.2 条目列表（itemize 环境）

条目列表是一种未编号的无序列表，它与编号列表的区别仅在于列表中每一项目的标号不一样，条目列表的标号是黑点、短横线和星号而不是编号形式。一个简单的条目列表可用下面的 itemize 环境

```
\begin{itemize} 列表项目 \end{itemize}
```

来创建。表 3.10 中列出了控制条目列表环境的命令。

要获得与默认定义不同形式的标号，就需要重新定义表 3.10 中列出的某些命令。例

表 3.10 控制条目列表环境的命令

	第 1 层	第 2 层	第 3 层	第 4 层
命令	\labelitemi	\labelitemii	\labelitemiii	\labelitemiv
定义	\m@th\bullet	\bfseries - -	\m@th\ast	\m@th\cdot
形式	•	–	*	·

如下面这个例子中我们利用 pifont 宏包中的符号将条目列表的第一层项目的标号定义为指向右边的手：

```
\renewcommand{\labelitemi}{\ding{43}}
\begin{itemize}
\item First item of the list.
\item Second item of the list.
\end{itemize}
```

- ☞ First item of the list.
- ☞ Second item of the list.

3.15.3 描述列表（description 环境）

描述列表或称定义列表，它的每一项目的标号不是编号也不是黑点等符号，而是任意一个我们要对之进行描述和定义的词语。这种形式犹如字典的条目，因而描述列表也称为字典列表。LaTeX 用 description 环境

```
\begin{description} 列表项目 \end{description}
```

来创建一个描述列表，其中列表项目的每一项之前都必须用 \item[词语] 来引导。这里词语就是我们要对之描述或定义的词语，它将被用黑体字体排印出来。由于被描述的词语有长有短，因此这种“标号”就不像编号列表和条目列表的标号那样整齐。我们可以通过修改控制 description 环境的命令 \descriptionlabel 来改变“标号”的字体。例如，下面 description 环境中的“标号”的字体被改成 sans serif 字体：

```
\renewcommand{\descriptionlabel}[1]%
 {\hspace{\labelsep}\textsf{#1}}
\begin{description}
\item[A.] First item of the list.
\item[B.] Second item of the list.
\end{description}
```

A. First item of the list.
B. Second item of the list.

3.15.4 自己定义列表

虽然 LaTeX 系统已经定义了几种列表环境，但是在使用中还是会遇到一些的列表结构不能用它们来排版，因而有必要自己定义所需要的列表环境。事实上，在 LaTeX 中诸如 enumerate, itemize, description 等列表环境，都可以用下面这个更一般的 list 环境

```
\begin{list}{默认标号}{列表声明}　列表项目　\end{list}
```

来定义，其中列表项目就是我们要罗列的一些项目，默认标号就是在没有给出 list 环境中命令 \item 的可选参数的情况下，加在每一个列表项目之前的标号，列表声明中包含一些不同的 list 环境几何参数值的设定。图 3.3 中显示了所有控制 list 环境的几何参数，它们可以分为垂直距离参数和水平距离参数两类。这些参数的值都可以用 \setlength 和 \addtolength 这两条命令来重新定义。

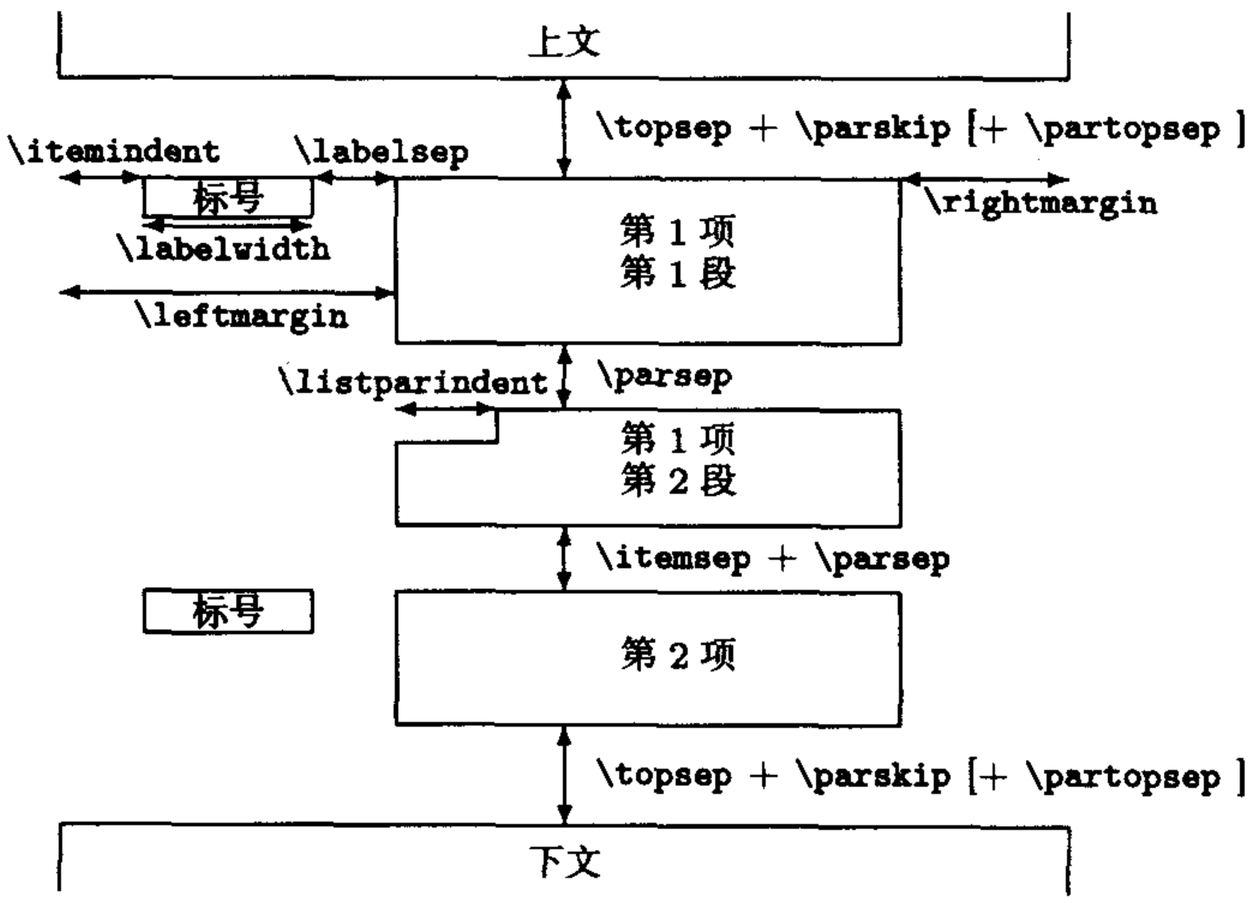


图 3.3　一般列表结构及参数

垂直距离参数包括：

\topsep　本层列表的第一项到上层列表或上文之间的垂直弹性距离。

\partopsep　在列表项里开始一个新段落时，添加到 \topsep 上的额外垂直弹性距离。

\parsep　在列表的一个单项里，段与段之间的垂直弹性距离。

\itemsep　在列表的项与项之间，添加到 \parsep 的垂直弹性距离。

水平距离参数包括：

\leftmargin	上层列表项的左边缘到本层列表项的左边缘之间的水平距离。它的值不能为负。
\rightmargin	上层列表项右边缘到本层列表项的右边缘之间的水平距离。这个参数的值通常是 0pt。
\listparindent	在列表的一个单项里，从第二段开始的段落首行缩格。其值可以取负数，但通常为 0pt。
\itemindent	列表项前面的标号到上层列表项左边缘的距离，通常为 0pt。
\labelwidth	标号所在盒子的宽度。其值如果大于或等于标号的自然宽度，那么标号在此盒子中是居右的。
\labelsep	标号所在盒子的右边缘与列表项之间的距离，其默认值是 0.5em。

默认标号可以是通常字符也可以是数学符号，但数学符号必须夹在两个美元符号 $ 之间。如果要对列表项目进行顺序编号，那么必须先在 list 环境之前用命令 \newcounter{*name*} 来定义一个计数器，其中 *name* 是计数器名。然后在列表声明部分用命令 \usecounter{*name*} 来调用名为 *name* 的计数器。计数器一旦被调用就可以在默认标号中使用打印计数器的命令如 \arabic{*name*}, \Roman{*name*} 等。例如：

```
\newcounter{myctr}
\begin{list}{\textbf{Example
\arabic{myctr}.}}{\usecounter{myctr}}
\item First item in this list.
\item Second item in this list.
\item Third item in this list.
\end{list}
```

Example 1. First item in this list.
Example 2. Second item in this list.
Example 3. Third item in this list.

此例中，先定义了一个新的计数器“myctr”，然后在 list 环境中把默认标号定义成“**Example 1.**”的形式，其中的阿拉伯数字用来对项目按顺序编号，因此要在列表声明中调用所定义的计数器。

在 LaTeX 中除了 enumerate, itemize 和 description 这三个列表环境之外还有一些环境（如 quote, quotation, center, flushleft 和 flushright 环境）也可以用 list 环境来定义。这些环境中只包含一个项目，而且项目的标号（是空标号）实际上是包含在定义中的。例如我们可以使用下面的语句

```
\newenvironment{Quote}{%
    \begin{list}{}{\setlength{\rightmargin}%
        {\leftmargin}}\item[]``\ignorespaces}{\unskip''%
```

```
\end{list}}
```

定义一个 Quote 环境，它类似于标准的 quote 环境，只是这里定义的环境中的文字被双引号括起来。注意，定义中命令 \ignorespaces 将忽略命令之后的空白，而命令 \unskip 忽略命令前面的空白。这使得双引号与括号中的文字不可分割。例如：

```
\ldots text before
\begin{Quote}
These are some text quoted by double
quote symbols.
\end{Quote}
sone text after the ``Quote''
environment.
```

…text before

> “These are some text quoted by double quote symbols.”

sone text after the “Quote” environment.

由于在 description 环境中，标号的宽度是由被描述的词语的宽度决定的而被描述的词语可长可短，因此描述语句的第一个词一般不能对齐。我们可以用下面的代码：

```
\newenvironment{Ventry}[1]%
  {\begin{list}{}{\renewcommand{\makelabel}[1]{\textsf{##1}\hfil}%
    \settowidth{\labelwidth}{\textsf{#1}}%
    \setlength{\labelsep}{5mm}
    \setlength{\leftmargin}{\labelwidth+\labelsep}}}%
  {\end{list}}
```

定义一个带参数的描述环境 Ventry，在这个环境中标号的宽度是由参数中的词语的宽度决定的。一般情况下，参数中的语句选择那个最长的被描述词语，而且参数中的语句的字体和大小最好与被描述词语的字体和大小一样，这样才能使每个条目都对齐。下面就是一个使用此环境的例子：

```
\begin{Ventry}{Return values}
\item[Description] Returns from a
function. If issued at tip-level,
the interpreter simply terminates,
just as if end of input had been
reached.
\item[Errors] None.
\item[Return values] Any arguments in
effect are\\ passed back to the caller.
\end{Ventry}
```

Description	Returns from a function. If issued at tip-level, the interpreter simply terminates, just as if end of input had been reached.
Errors	None.
Return values	Any arguments in effect are passed back to the caller.

在这个例子中我们使用了命令 \makelabel 。这是一条 LaTeX 内部命令，列表环

境中的每一个 \item 命令都会调用 \makelabel 来产生一个标号。因此可以重新定义 \makelabel 以使标号具有不同的形式，但对它的修改只能在列表声明中进行，这是因为一进入列表环境，内部的 \makelabel 就被启用，从而覆盖了环境之外对它所做的修改。另外，这里的 Ventry 环境以及命令 \makelabel 本身都有一个参数，因此重新定义 \makelabel 时必须用 ##1 而不是 #1 ，这样才能不使两种参数混淆。

3.16 章节标题的结构

不同的文件类型有不同的章节标题结构。在实际使用中经常出现已有的文件类型（如 book、article 等）的章节标题结构不符合要求的情况，此时就需要用户自己来定义了。

3.16.1 节序号的修改

在 LaTeX 中，自动编排的序号都是由计数器来完成的。指向章节序号的计数器是 chapter、section、subsection 等，因此 \thechapter、\thesection、\thesubsection 等就分别代表它们的序号。比如下面的命令

```
\renewcommand{\thechapter}{\arabic{chapter}}
\renewcommand{\thesection}{\thechapter.\arabic{section}}
\renewcommand{\thesubsection}{\thesection.\arabic{subsection}}
```

用阿拉伯数字表示章、节和小节的序号，但节序号前面带有章序号，小节前带有章和节的序号。下面的命令

```
\renewcommand{\thechapter}{\Alph{chapter}}
```

把章序号改为用大写字母来表示。不过用这种方法并不能任意修改序号的形式。比如要把标题中的节序号用方框框起来，而文本中的节序号保持不变，就不能用下面的命令

```
\renewcommand{\thesection}{\fbox{\thechapter.\arabic{section}}}
```

来修改，这是因为这样的定义使得在其他地方进行交叉引用时的节序号也都带有方框，从而使版面显得不美观。事实上，在标准 LaTeX 中定义节序号形式的是一个内部命令 \@seccntformat，它大体上与

```
\csname the#1\endcsname\hspace{0.5em}
```

相当，其中 #1 就是相应的序号。因此按如下方式修改这条命令就可以达到要求：

```
\makeatletter
\renewcommand{\@seccntformat}[1]{\fbox%
{\csname the#1\endcsname}\hspace{0.5em}}
\makeatother
\section{This is correct}\label{sec:OK}
Referencing a section using this
definition generates the correct result
for the section reference~\ref{sec:OK}.
```

4.7 This is correct

Referencing a section using this definition generates the correct result for the section reference 4.7.

如果在上面这个例子中将 `\fbox{\csname the#1\endcsname}` 改为

```
\S\csname the#1\endcsname
```

则可以在节序号前面加上符号 §。

注意，对命令 `\@seccntformat` 的修改，会影响所有用 `\@startsection` 命令定义的标题（如 section、subsection 等）。因此，要使不同层次的标题具有不同的形式还必须（按下一小节叙述的方法）修改不同层次节标题结构的定义。

3.16.2 修改节标题形式

LaTeX 提供了一条很一般的命令 `\@startsection` 用来定义节标题的形式。要定义或者修改节标题命令就得看一下 `\@startsection` 到底能干些什么。如果这条命令的所有功能还不能满足我们的要求，那么可以使用功能更强大的 `\secdef` 命令，用它可以任意改变章节标题的形式。

粗略地说，节标题可分为两种：独立显示形式和段内显示形式。独立显示的标题是单独一个段落，并距上下文有一定的距离。而段内显示的标题与上文有一定距离，但与下文在同一个段落。两种形式的节标题命令都可以用下面的命令

`\@startsection{`*name*`}{`*level*`}{`*indent*`}{`*beforeskip*`}{`*afterskip*`}{`*style*`}`

来定义。其中

name 表示所定义的节标题命令的名称（不带反斜杠），比如 section 和 subsection 等。

level 是一个数，表示所定义的节标题命令的层次。这个数决定了定义的节标题是否被编号（若它小于或等于 `secnumdepth` 则被编号，见 3.8.3 节），也决定了标题是否会被编进目录（若它小于或等于 `tocdepth` 则被编进目录，见 3.9.1 节）。

indent 是一个长度，表示定义的节标题到版心左边缘的距离。此度量若是负的，则标题会进入边空。

beforeskip 是一个长度，其绝对值表示标题到上文之间的距离。若此度量是负的，则标题后面的第一个段落不缩格。此度量最好是一个可以被伸长和缩短的弹性长度。另外，标题总是另起一段的，因而 \parskip 已被加进标题与上文之间的距离了。

afterskip 是一个长度，其绝对值表示独立显示的标题到上文之间的垂直距离或者段内显示的标题到下文之间的水平距离。若此度量是负的，则定义的标题是段内显示的。对于独立显示的标题， \parskip 已被加进标题与下文之间的距离了。

style 它决定标题内容的形式。可以是任何影响文本排版的结构，如选择字体尺寸的命令 \Large、\bfseries, 左对齐命令 \raggedright, 居中命令 \centering 等。

图 3.4 和图 3.5 分别指明了独立显示和段内显示的标题所用的参数。

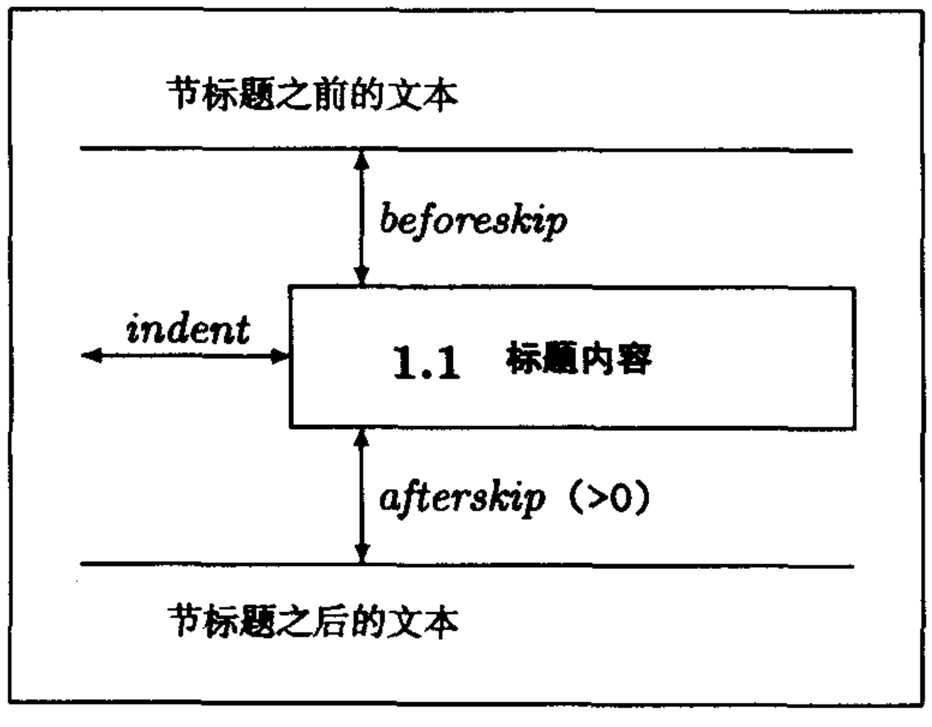


图 3.4 独立显示的节标题

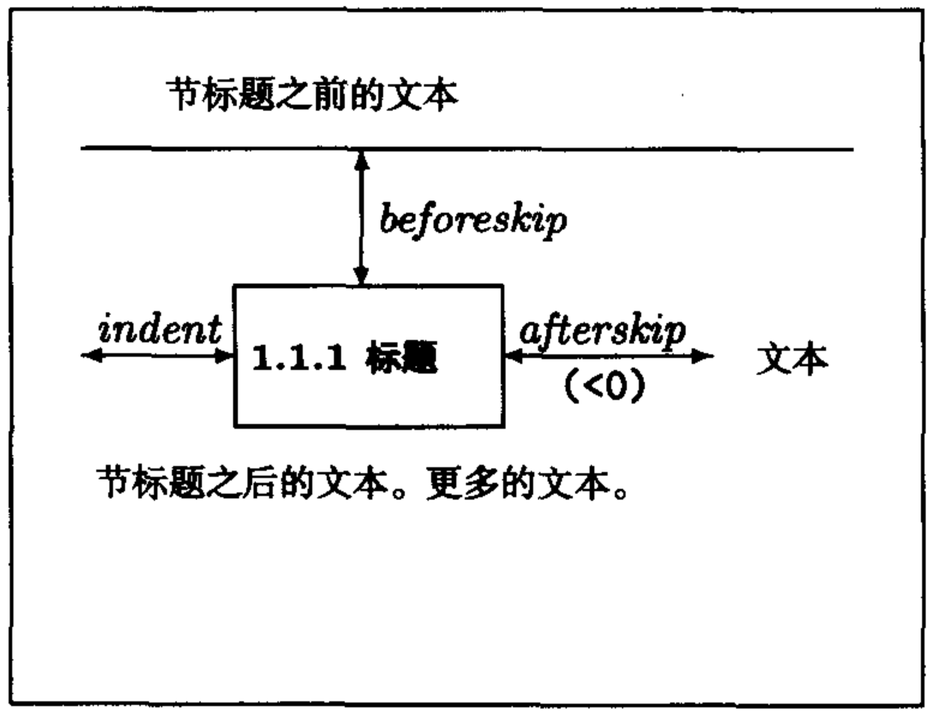


图 3.5 段内显示的节标题

3.16.3 修改章标题形式

由于 \part 和 \chapter 这两个标题命令并不是用 \@startsection 来定义的，因此不能用上面的方法来修改。但可以用 \secdef 这条更一般的命令修改。比如修改 \chapter 命令，可以用类似于：

```
\renewcommand{\chapter}{…\secdef \cmda \cmdb }
\newcommand{\cmda}[2][default]{ … }
\newcommand{\cmdb}[1]{ … }
```

这样的定义，其中定义了两条新命令 \cmda 和 \cmdb。于是

```
\chapter{标题}                    将用命令 \cmda 确定 标题
```

```
\chapter[目录标题]{标题}        将用命令  \cmda 确定 目录标题
\chapter*{标题}                  将用命令  \cmdb 确定 标题
```

命令 \secdef 可以任意对标题的形式进行修改，但是这条命令使用起来也很难，因此一般是在编写宏包时才使用。实际上也可以用命令 \@makechapterhead 和 \@makeschapterhead 来适当地修改章标题的形式（前一个是针对 \chapter 命令而后一个针对 \chapter* 命令）。例如，用 CJK 宏包排版中文稿件时可以对章标题的形式做如下修改：

```
\makelatletter
\renewcommand{\chaptername}{第~\CJKnumber{\thechapter}~章}
\renewcommand{\@makechapterhead}[1]{%
    \vspace*{-\baselineskip}%
    {\normalfont \centering\Large\bfseries%
    \chaptername \quad #1 \par\nobreak%
    \vspace{1.5\baselineskip}
    }}
\renewcommand{\@makeschapterhead}[1]{%
    \vspace*{-\baselineskip}%
    {\normalfont \centering\Large\bfseries  #1 \par\nobreak%
    \vspace{1.5\baselineskip}
    }}
\makeatother
```

所得到的章标题的形式与本书相同，只是“第 1 章”改成“第一章”。

一般情况下，每章标题后面的第一个段落是不缩格的。如果希望它也有缩格，那么可以按如下方法

```
\makelatletter
\renewcommand{\@afterheading}{%
    \@nobreaktrue
    \everypar{%
    \if@nobreak
    \@nobreakfalse
    \clubpenalty \@M
    \else
    \clubpenalty \@clubpenalty
    \everypar{}%
    \fi}}
\makeatother
```

修改命令 \@afterheading。当然也可以用 indentfirst 宏包来使首行缩格（见 3.2.3 节）。

这一节所介绍的修改章节结构的命令中有许多地方出现符号 @，它一般是用在宏包的定义中。源文件中要使用这种带 @ 的命令，必须将命令放在命令 \makeatletter 和 \makeatother 之中。由于它们是对类型文件或者宏包中原来的定义直接进行修改，所以对 TeX 或者 LaTeX 命令不熟悉的读者会觉得上面所述的方法不容易理解。下一节中我们介绍一些面向最终用户的高端命令用来对章节结构、页面结构以及目录结构进行全面设置。

3.17 设置章节、页面和目录结构

Javier Bezos 编写的 titlesec 和 titletoc 是两个功能强大的宏包，用它们可以对页面结构、章节结构和目录结构进行全面设置。这里只叙述其精要，若想进一步深入了解这两个宏包的内容，可以查阅它们的说明文件。

3.17.1 章节标题结构

要使用 titlesec 宏包设置章节的标题结构，必须在源文件的导言部分使用命令：

```
\usepackage[选项]{titlesec}
```

其中的选项有许多选择，以下列举一部分：

rm sf tt md bf up it sl sc 这些选项用来设置标题的字体，对所有标题（包括章、节、小节等）都起作用。默认值是 bf。

big medium small tiny 这些选项用来设置标题字体的尺寸，对所有标题都起作用。默认值是 big。标题尺寸会按层次顺序递减。

raggedleft center raggedright 设置标题为居右、居中、居左。

indentafter noindentafter 设置标题下面第一个段落是否缩格。这个选项对排版中文特别有用。有了这个选项就不需要再用 indentfirst 宏包了。

以下介绍设置章节结构的命令。这些命令（包括设置目录的命令）可以在导言部分使用也可以在正文部分使用。但是若放在一个局部环境中，则它们对环境之外的内容不起作用。这一点在用 CJK 排版中文稿件时很重要，也就是说不能将它们放在导言部分的 CJK 环境中。当使用 CJK 时，若导言部分含有中文，则只需在设置的前面放一个空的 CJK 环境，如：

```
\begin{CJK}{GBK}{st}        \end{CJK}
```

先介绍用来设置章节标题的命令：

```
\titleformat{命令}[形式]{格式}{标题头定义}{间距}{前命令}[后命令]
```

其中

命令　就是排版标题的命令，如 \part、\chapter、\section 等。

形式　表示标题的结构形式，它可以是如下的值：

- hang　这是默认值。表示标题头与标题内容在同一行。
- block　将整个标题放在小页中。这对于居中的标题以及某些特别的（如包含图形的）标题很有用。
- display　将标题头和标题内容放在不同的段落。标题头在前标题内容在后。
- runin　设置段内标题。
- leftmargin　将标题放在左边空里。
- rightmargin　将标题放在右边空里。
- frame　带边框的标题。标题头在边框线上。
- wrap　将标题放在左边，并且第一个段落对它进行绕排。第一个段落必须比标题长。

格式　就是标题的排版格式。可以在这里用一些水平和垂直意义上的命令，它们在标题前的垂直空白之后被执行。如 \large、\bfseries、\vspace、\centering、\titlerule 等命令都可以在这里使用。

标题头定义　即如何定义标题头。如 \thesection。若此项空置，则无标题头。

间距　是标题头与标题内容之间的空白长度。

前命令　是一些在排版标题之前先执行的命令。常空置。

后命令　是一些在排版标题之后再执行的命令。经常不给出。

```
\titleformat*{命令}{格式}
```

这条命令是 \titleformat 的缩略形式，它只用于修改标题的格式。

```
\titlelabel{标题头定义}
```

这条命令用于修改标题头定义。但如果已经使用 \titleformat，那么这条命令就不起作用了。

```
\titlespacing*{命令}{左距离}{上距离}{下距离}[右距离]
```

这条命令设置标题与四周的间距。其中

命令 是排版标题的命令，如 \part, \chapter, \section 等。

左距离 是标题到版心左边缘的距离，用它设置标题的宽度。负距离会使标题超过左边缘。若在 \titleformat 中使用了 leftmargin、rightmargin 和 drop 形式选项，那么这里的设置也不起作用。

上距离 是标题到上文之间的垂直距离。

下距离 是标题到下文之间的距离。对于 hang、block 和 display 选项，它是垂直距离。对于 runin、drop、wrap、leftmargin 和 rightmargin 形式选项，它是水平距离。

右距离 当使用 hang、block 和 display 选项时，这个选项能增加标题到右边缘的距离。

如果在 \titlespacing* 中，对上距离和下距离使用弹性距离，而又不想用 plus 和 minus 这样的参数，那么可以用不带星号的命令

\titlespacing{命令}{左距离}{*n}{*n}

其中命令和左距离的意义同上。n 是正十进制数。对于上距离来说，*n 表示：$n$ex plus .3ex minus .06ex，对于下距离来说，*n 表示：$n$ex plus .1ex。

下面是几条在 \titleformat 和 \titlespacing 中使用的与空白有关的命令：

\filright \filcenter \filleft \fillast \filinner \filouter

这几条命令类似于 \raggedright 等命令，但稍有不同。特别是，\raggedright 等命令会删除由 \titlespacing 设置的左边空白和右边空白。\fillast 对齐段落的两边，但最后一行居中。这几条命令也可以在带边框的标题中使用。

\filinner 和 \filouter 实际上分别是 \filleft 或 \filright，但依赖于奇数页还是偶数页。由于 TeX 断页的不同步，这两条命令只能用于 \chapter 的定义中时才起作用。

\chaptertitlename

这条命令一般情况下与 \chaptername 相同，但在附录中它与 \appendixname 等同。

\titleline[位置]{水平素材}

这条命令用于 \titleformat 中，它允许在标题中添加一行内容。水平素材可以是文本和其他内容，位置是水平素材在这一行的水平位置（可取 l，c，r）。实际上这条命令先创建一个宽为 \linewidth 水平盒子，然后将水平素材置于盒子中。当水平素材的宽度小于 \linewidth 时，最好使用位置选项，否则会出现“underfull hbox”的错误信

息。此命令通常用于在标题中画线或书写标题引子。

\titlerule{*height*}
\titlerule*[*width*]{*text*}

是两条在 \titleformat 中用来画线的命令。*height* 和 *width* 都是长度参数。其中第一条命令画一条粗为 *height*，长为 \linewidth 的线，第二条带星号的命令将 *text* 放在宽为 *width* 的盒子中，并连续打印这个盒子使其占满一行。下面给出了一个设置章标题的例子：

```
\titlespacing*{\chapter}{0pt}%
{-\baselineskip}{1.5\baselineskip}
\titleformat{\chapter}[display]%
{\titlerule\Large\bfseries\centering}%
{\flushleft 第~\thechapter~章}%
{1em}{}[\vspace{2mm}\titlerule]
```

第 1 章

函数的连续性

3.17.2 页版式设计

在 3.5.4 节中，我们介绍了如何用 fancyhdr 宏包来设计页眉和页脚。为了完整性，titlesec 宏包也提供了一些命令用来排版页眉和页脚。

\newpagestyle{*name*}[*global-style*]{*commands*}
\renewpagestyle{*name*}[*global-style*]{*commands*}

这两条命令分别用来定义新的和修改已有的页版式命令，为了方便起见，定义中行末的空格被自动取消，因此无需在每行的行末加百分号。定义中的 *name* 就是定义的页版式名称，*global-style* 是任何同时作用于页眉和页脚的命令，*comamnds* 可以是下面一些用来定义页版式的命令：

\headrule	\setheadrule{*length*}
\footrule	\setfootrule{*length*}

这几条命令用来设置页眉线和页脚线。其中右边两条命令设置线条的粗细，左边两条命令输出线条。比如 \setheadrule{.4pt} 将页眉线设置成 0.4pt 粗，这也是默认值。也可以通过修改下面的命令来设置更一般的页眉线和页脚线：

\makeheadrule	\makefootrule

titlesec 内部就是用这两条命令来设置页眉线和页脚线的，比如 \setheadrule{0.8pt} 等价于用下面的命令

```
\renewcommand{\makeheadrule}{%
    \rule[-.3\baselineskip]{\linewidth}{0.8pt}}
```

进行的设置。由于 \makeheadrule 这条命令的基线与页眉的基线是重合的，因此它包含的画线命令 \rule 应将线条往下移动一些。比如上面的命令将线条下移了 .3\baselineskip。再比如用下面的方法可以定义画两条页眉线的命令，上面的线粗 0.8pt，下面的线粗 0.4pt：

```
\renewcommand{\makeheadrule}{%
    \makebox[0pt][l]{\rule[-.3\baselineskip]{\linewidth}{0.8pt}}%
    \rule[-.45\baselineskip]{\linewidth}{0.4pt}}%
```

下面的两条命令用来设置页眉和页脚：

```
\sethead[偶数页左边页眉][偶数页中间页眉][偶数页右边页眉]
        {奇数页左边页眉}{奇数页中间页眉}{奇数页右边页眉}
\setfoot[偶数页左边页脚][偶数页中间页脚][偶数页右边页脚]
        {奇数页左边页脚}{奇数页中间页脚}{奇数页右边页脚}
```

若偶数页的页眉和页脚不给出，则表明与奇数页的设置一样。在这两条命令中一般可以使用下面的一些命令：

\thechapter, \thesection … 打印章节等的编号。

Comchaptertitle, \sectiontitle … 打印章节等的标题。

\ifthechapter{*true*}{*false*}, \ifthesection{*true*}{*false*} … 这些检测条件语句会输出 *true*，除非缺少标题头或者正好处于某个中间层次标题还未产生（如在 \chapter 与它后面第一个 \section 之间）。

\thepage 打印页码。

比如本书的页眉页脚格式可以按如下方法设置：

```
\newpagestyle{mystyle}{%
\renewcommand{\makeheadrule}{%
\makebox[0pt][l]{\rule[-.3\baselineskip]{\linewidth}{0.8pt}}%
\rule[-.45\baselineskip]{\linewidth}{0.4pt}}%
\sethead[\thepage][][第~\thechapter~章\hspace{1em}\chaptertitle]%
{\thesection\hspace{1em}\sectiontitle}{}{\thepage}
\setfoot[][][]{}{}{}
\pagestyle{mystyle}
```

这里是先定义一个名为 mystyle 的页版式，然后使用这个页版式。

```
\setmarks{第 1 标志}{第 2 标志}
```

这条命令设置使用哪个层次的标题作为标志（mark），以及标志如何更新。例如

```
\setmarks{chapter}{section}
```

的意义是：

- 分别使用 \chaptertitle 和 \sectiontitle 作为第 1 标志和第 2 标志。
- \sectiontitle 隶属于 \chapter。
- 可以用 \ifthechapter 和 \ifthesection 这两个检测条件语句。
- 标志随着 \chapter 和 \section 更新。

对于 book 类型的文件，\setmarks{chapter}{section} 是默认设置；但对于 article 类型的文件，默认设置是 \setmarks{section}{subsection}。\setmarks 命令可以在定义或者更新页版式的命令 \(re)newpagestyle 之外使用，但最好在 \pagestyle 之前，否则不起作用。

如果使用了 \setmarks 命令，那么就不需要再用标准 LaTeX 中的 \markboth 命令了。因为它是用来设置 myheadings 页版式的，在这里没有意义。

另外，还可以使用下面的命令来设置比版心更宽的页眉和页脚：

```
\widenhead[偶数页左增量][偶数页右增量]{奇数页左增量}{奇数页右增量}
```

这条命令将页眉及页脚的两边分别增加一个增量。比如命令

```
\widenhead[1cm][2cm]{1cm}{2cm}
```

将每页的页眉和页脚左边伸长 1 厘米，右边伸长 2 厘米。

3.17.3 目录结构

伴随着 titlesec 宏包，Javier Bezos 还编写了一个用来设置目录结构的 titletoc 宏包。这里只介绍几个主要的命令。

```
\titlecontents{章节名称}[左距离]{上部命令}{标题头及前部命令}
                {前部命令}{填充符号及页码}[后部命令]
```

这是设置目录结构的命令，其中

章节名称 是不含反斜杠的章节的名称，如 part, chapter, section, subsection 等。

左距离 是目录中标题内容（不是标题头）到正文左边缘的距离。

上部命令 是对标题整体的格式化，可以是一些垂直素材。通常在这里设置标题的字体、到上面的距离等。也可以在这里添加文字，添加的文字会被加到标题内容的前面。

标题头及前部命令 是包含标题头定义及其在它前面可以使用的一些命令。也可以直接在这里添加文字，添加的文字也会被加到标题内容的前面。

前部命令 是标题内容（不含标题头）前面可以使用的一些命令。常空置。

填充命令及页码 是排印页码及设置页码与标题内容之间的填充符号（填充符号通常用小圆点）的命令。

后部命令 是标题后面可以使用的一些命令。可以在这里设置垂直距离。

下面两条命令

```
\thecontentslabel 和 \thecontentspage
```

分别表示目录中的标题头（通常是标题的序号）和页码。

下面的命令

```
\contentslabel[格式]{宽度}
```

用于 \titlecontents 中，来设置目录中的标题头。其中格式是标题头的格式化命令，宽度是标题头所在盒子的宽度。

下面的命令

```
\contentspage[格式]
```

用来设置目录中页码的格式。若格式选项不给出，则用通常的阿拉伯数字排印页码。注意当页码的数字较大是黑体时（如章的页码），有可能出现 Overfull hbox 的错误信息，此时有必要使用格式选项。

下面我们给出一个（结合 CJK 宏包）设置目录结构的例子，其输出效果与本书目录结构类似。

```
\titlecontents{chapter}
    [0mm]
    {\vspace{\baselineskip}\bfseries}
    {第~\thecontentslabel~章\hspace{1.1em}}
    {}
    {\hfill\contentspage[{\makebox[0pt][r]{\thecontentspage}}]}
\titlecontents{section}
    [15mm]
    {\vspace{.2\baselineskip}}
    {\contentslabel[\thecontentslabel]{10mm}}
    {}
    {\titlerule*[1pc]{$\cdot$}\contentspage}
\titlecontents{subsection}
```

```
[1.5cm]
{\vspace{.2\baselineskip}}
{\thecontentslabel\hspace{1em}}
{}
{\titlerule*[1pc]{$\cdot$}\contentspage}
```

第 4 章　表　　格

表格是处理数据最常用的一种形式，也是印刷出版物中一个基本设计单元。表格不但形式简洁，而且反映了行与列中数据之间的关系，使得对比分析更容易理解。在 LaTeX 系统中，有两种方法将数据以行与列的形式整齐地排列，一种方法是利用 tabbing 环境，另一种方法是利用 tabular 环境和 array 环境。

4.1　tabbing 环境

在用打字机打印文本或用编辑软件编辑文本文件时，只要设置好 Tab 间距就可以利用键盘上的 Tab 键将数据按行列方式自然对齐。LaTeX 中 tabbing 环境的功能与此类似。例如：

```
\begin{tabbing}
Material\quad\= Quality\quad\=
Color\quad\= Price\\[0.8ex]
Paper \> medium \> white \> low\\
Leather \> good \> brown \> high\\
Card \> bad \> gray \> medium
\end{tabbing}
```

Material	Quality	Color	Price
Paper	medium	white	low
Leather	good	brown	high
Card	bad	gray	medium

注意在 tabbing 环境中的第 1 行必须用 \= 来设置 Tab 站点，而在后续行中用 \> 来指明跳过一个 Tab 站点。无论第 1 行还是后续行都用 \\ 来换行。

如果用命令 \kill 来取代 tabbing 环境中第 1 行的换行符 \\ ，则该行的内容就可以不显示出来，只是作为一个样本行设置了 Tab 站点。如下面的例子使用了 \hspace 命令来设置每一列的宽度。

```
\begin{tabbing}
\hspace{1.6cm}\= \hspace{1.8cm}\=
\hspace{1.5cm}\= \hspace{1.5cm}\kill
Paper \> medium \> white \> low\\
Leather \> good \> brown \> high\\
Card \> bad \> gray \> medium
\end{tabbing}
```

Paper	medium	white	low
Leather	good	brown	high
Card	bad	gray	medium

还可以在 tabbing 环境中最后一列的文字前面加上 \‘ 使最后一列的文字靠右对齐。另外可以在除第一行以外的其他行中重新设置 Tab 站点。

在第 18 页的 2.12.5 节中，已说明命令 \=、\' 和 \` 是用来表示字母的重音的，但在 tabbing 环境中，这些命令的意义已被重新定义。若要使用它们以前的定义，可以用 \a=、\a' 和 \a` 分别取代之。

在 tabbing 环境中使用命令 *left_text* \' *right_text* 可以将 *left_text* 排印在它前面的 Tab 站点的左边，到 Tab 站点有确定的距离，且从 Tab 站点开始将 *right_text* 排印在右边。*left_text* 到 Tab 站点的距离由长度参数 \tabbingsep 控制，它的默认值与 \labelsep 相等，通常是 5pt。下面的例子已将 \tabbingsep 设置为 3ex。

```
\setlength{\tabbingsep}{3ex}
\begin{tabbing}
\hspace{2.8cm} \= \kill
\> Beethoven\' Symphony no.6 in F\\
\> Rachmaninov\' Concerto for Piano\\
\> Wagner\' Tannhauser
\end{tabbing}
```

Beethoven Symphony no.6 in F
Rachmaninov Concerto for Piano
Wagner Tannhauser

注意 LaTeX 将 tabbing 环境中的文本当作通常的段落来对待，因此有可能自动把某一行以后的内容排印到下一页。但是不可以在 tabbing 环境中使用 \newpage, \clearpage, \pagebreak 等强行换页命令。如果想把某一行开始的内容排到下一页，可以在这一行的前面插入一个充分大的垂直空白距离。

4.2 tabular 环境和 array 环境

前面所述的 tabbing 环境可以创建各式各样的无框表格，如果与盒子结合起来使用也可以创建带框的表格，但做起来十分复杂。因此 LaTeX 提供了下面的 tabular, tabular* 和 array 环境，利用这些环境就能很方便地创建各种有框或无框表格了。

```
\begin{tabular}[位置]{列格式} 表格行 \end{tabular}
\begin{tabular*}{宽度}[位置]{列格式} 表格行 \end{tabular*}
\begin{array}[位置]{列格式} 表格行 \end{array}
```

其中 array 环境中各种参数的意义及使用方法与 tabular 环境相同，但 array 环境只能用在数学模式中。以下详细介绍 tabular 环境中的参数意义及使用方法。

位置　表示表格与表格之外文字在垂直方向上的对齐方式。其意义与 \parbox 的位置参数类似，可以取 t, c 和 b 这三个选项。其中

t　指表格第 1 行的基线或上边框与表格之外当前行文字的基线对齐。

c 指表格中间与表格之外当前行文字的基线对齐。这也是默认参数。

b 指表格末行的基线或下边框与表格之外当前行文字的基线对齐。

列格式 指示表格中各列按何种方式排列，如左对齐、右对齐、居中等。必须对每一列都进行说明。在列格式中可以使用如下选项：

l 表示对应列按左对齐方式排列。

c 表示对应列按居中方式排列。

r 表示对应列按右对齐方式排列。

p{*width*} 表示对应列中的文本被排在宽度为 *width* 的段落盒子中，且盒子中文本的第 1 行与其他列中相应的行对齐。这相当于在此列中每一行都使用了 \parbox[t]{*width*}{列文本}。

| 表示在表格中画一条垂直线。

|| 表示在表格中画两条紧靠的垂直线，两条垂直线的间距可用长度参数来 \doublerulesep 设置。

*{*num*}{*cols*} 表示 *cols* 被重复了 *num* 次，其中 *num* 是正整数，*cols* 是其他列格式选项。例如 *{4}{|c}| 代表 |c|c|c|c|，表示表格有 4 列，表中每个元素在列里都是居中的，而且每列的两边都有一条垂直的表格线。

@{文本} 这种列格式选项被称为 @-表达式，它表示忽略两旁的列之间的空白，并在每一行中插入文本。由于 @-表达式两旁的列之间的空白已被删除，要想使用 @-表达式并且留有一定空白，就必须在 @-表达式中使用 \hspace 命令。另外，在 @-表达式中也可以使用 \extracolsep{*width*}，它指明在此之后而在下一条命令 \extracolsep 之前的所有列之间都插入宽度为 *width* 的空白，这种空白不会被 @-表达式删除。通常表格的第 1 列与表格左边缘，以及最后一列与表格右边缘之间会有少许空白，如果不需要这种空白，可以在列格式选项的最前面和最后面使用不含内容的 @-表达式，即 @{}。对于 tabular* 环境，为了使表格能被伸展到指定的宽度，就必须在列格式参数中的某处使用 @{\extracolsep{\fill}}。

表格行 就是表格中各行文本数据。每一行都要用换行符号 \\ 来结束。在每一行中居于不同列的文本必须用符号 & 分开。通常情况下，如果表格有 n 列，那么每一行应包含 $n-1$ 个 &，除非在行中使用了 \multicolumn 命令，此时文本数据将占有多于一列的空间。在表格行中或行与行之间可以使用如下一些命令：

\hline 此命令只能放置于两行之间，即第 1 行之前或紧跟在某个换行符 \\ 之后，它指示画一条与表格同样宽度的水平直线。连续两个 \hline 命令，表示画两条相隔少许间距的水平直线，这个间距可以用长度参数 \doublerulesep 来设置。

\cline{$n-m$} 此命令表示从第 n 列左边缘起至第 m 列右边缘结束画一条水平直线。同命令 \hline 一样，此命令也必须紧跟在换行符 \\ 之后。

\multicloum{n}{col}{文本} 此命令表示在本行中将后面的 n 列结合成一列（ n 是正整数），因而文本占据 n 列位置。col 是列格式参数，它可以取位置参数 l, c, r 之一，其中也可以包含 @-表达式和画竖线的符号 |。注意，此命令只能用于第 1 行的前面或者紧跟在某个 & 之后。

\vline 此命令表示在当前位置画一条与行等高的垂直线段。它不会增加整个表格的高度。

宽度 此参数只能用于 tabular* 环境，且不可省略，它表示整个表格的宽度。在此情形下，列格式 选项中某处必须包含 @{\extracolsep{\fill}} 这种所谓 @-表达式，这样才能把表格伸展到预定的宽度。

在创建表格时内在地使用了一些长度参数，这些长度参数的值是可以改变的，但改变这些参数值的命令只能置于源文件的导言部分或者表格之外，而不能置于表格之内。下面列出了这些长度参数命令：

\tabcolsep 它在用 tabular 和 tabular* 环境创建表格时，插入两列之间的空白宽度的一半。

\arraycolsep 用 array 环境创建表格时，插入两列之间的空白宽度的一半。

\arrayrulewidth 用来设置表格线的粗细。

\doublerulesep 用来设置表格双线之间的间距。

\arraystretch 用来改变表格中行与行之间的距离。它的默认值是 1，这是一个因子。若将其值改为 1.5，则行距是原先的 1.5 倍，即行距增加了 50%。

前面 4 个长度参数的值都可以用 \setlength 这条命令来修改。如命令

```
\setlength{\arrayrulewidth}{0.4pt}
```

将表格线的粗细设置成 0.4pt。第 5 个参数的值可以用命令 \renewcommand 重置。例如，命令

```
\renewcommand{\arraystretch}{n}
```

将表格之内的行距改成原先行距的 n 倍。

4.3 表格例子

在源文件中输入下面的内容

```
\begin{tabular}{|r|l||c|rrr|r@{:}l|c|}\hline
Position & Club & Games & W & T & L &\multicolumn{2}{c|}{Goals} & Points\\
\hline\hline 1 & Amesville Rockets & 33 & 19 & 13 & 1 & 66 & 31 &
51:15\\ \hline 2 & Borden Comets & 33 & 18 & 9 & 6 & 65 & 37 &
45:21\\ \hline 3 & Quincy Giants & 33 & 7 & 5 & 21 & 40 & 89 &
19:47\\ \hline 4 & Ralston Regulars & 33 & 3 & 11 & 19 & 37 & 74 &
17:49 \\ \hline
\end{tabular}
```

将排印出表格：

Position	Club	Games	W	T	L	Goals	Points
1	Amesville Rockets	33	19	13	1	66:31	51:15
2	Borden Comets	33	18	9	6	65:37	45:21
3	Quincy Giants	33	7	5	21	40:89	19:47
4	Ralston Regulars	33	3	11	19	37:74	17:49

这是一个有 9 列 5 行的带框表格。列格式选项中使用了 @-表达式将第 7 列和第 8 列紧靠在一起且中间插入冒号。第 1 行中的第 7 列和第 8 列被合并了，注意此时这两列之间的垂直线也同时被删除。因此，如果将 `\multicolumn{2}{c|}{Goals}` 中的 `c|` 换成 `c` 的话， Goals 后面就少一条垂直线。

在下面的例子中我们编排一个小学课程表。

```
\newcommand{\rb}[1]{\raisebox{1.5ex}[0pt]{#1}}
\begin{tabular}{|c|c|c|c|c|c|c|}\hline
\multicolumn{7}{|c|}{\rule[-3mm]{0mm}{8mm}\bfseries 课\quad
程\quad 表}\\\hline & 节次 & 星期一 & 星期二 & 星期三 & 星期四 &
星期五 \\ \hline & 1 & 语文 & & 数学 & 英语 & 数学 \\
\cline{2-3}\cline{5-7} \rb{上} & 2 & 英语 & \rb{数学} & 美术 &
数学 & 语文 \\ \cline{2-7} & 3 & 数学 & 语文 & & 语文 & 体育 \\
\cline{2-4}\cline{6-7} \rb{午} & 4 & & 常识 & \rb{语文} & & 美术
\\ \hline\hline 下 & 5 & 语文 & 语文 & 数学  & & 常识 \\
\cline{2-5}\cline{7-7} 午 & 6 & 音乐 & 体育 & 音乐 & \rb{作文} &
思品 \\ \hline
\end{tabular}
```

课 程 表						
	节次	星期一	星期二	星期三	星期四	星期五
上午	1	语文	数学	数学	英语	数学
	2	英语		美术	数学	语文
	3	数学	语文	语文	语文	体育
	4		常识			美术
下午	5	语文	语文	数学	作文	常识
	6	音乐	体育	音乐		思品

在此表格中的第 1 行使用了命令 \multicolumn 将 7 列合并成 1 列，并且用零宽度的标尺盒子 \rule[-3mm]{0mm}{8mm} 做了一个不可见的支柱（见 2.14.6 节），使第 1 行的高度增加到 8mm。在表格的前面还用 \raisebox{1.5ex}[0pt]{#1} 定义了带一个参数的新命令 \rb，它将其中的文本向上移动 1.5ex，这样就使格子中的文本看起来上下对称。另外，我们在第 3 行和第 4 行之间以及第 5 行和第 6 行之间都连续地使用了命令 \cline{*n*-*m*} 来画水平线段。

4.4 用 array 宏包增强 tabular 环境

虽然用标准 LaTeX 能排出很复杂的表格，但其功能还是有限，使用起来也不是很方便。为了使创建表格变得更容易，几年来相继出现了一些用于排版表格的宏包，如 array, multirow, hhline, longtable 等宏包。这些宏包中有的增加了标准 tabular 环境和 array 环境的选项，从而增强了排版表格的功能，有的定义了新的表格环境来排版特殊表格。在本小节以及后面几个小节中我们介绍若干常用的有关表格的宏包。

在 4.2 节中，我们已列出了 tabular 环境的所有列格式选项并说明了它们的意义。比如，列格式选项 p{*width*} 是指将相应的列中的数据分别排在一个宽度为 *width* 的段落盒子中，而且数据排在盒子的顶部。那么能不能将数据排在盒子的中部或底部呢？array 宏包（属于 Tools 宏包套件）提供了新的列格式选项 m{*width*} 和 b{*width*} 使得这一问题很容易得到解决。表 4.1 中列出了 array 宏包提供的 tabular 环境列格式选项，其中有的是旧的，有的是新的。

例如，在排版表 4.1 时我们就使用了列格式 {|c|m{9cm}|}。再例如，通过选项 > 和 < 还可以建立一个数学模式列：

表 4.1 array 宏包提供的 tabular 环境列格式选项

未变选项	说 明
l	原始 tabular 环境的选项，表示对应的列文本左对齐。
c	原始 tabular 环境的选项，表示对应的列文本居中。
r	原始 tabular 环境的选项，表示对应的列文本右对齐。
p{*width*}	原始 tabular 环境的选项，等同于 \parbox[t]{*width*}。
@{*text*}	原始 tabular 环境的选项，表示删除左右两列之间的空白并插入 *text*。
修正选项	**说 明**
|	表示在两列之间插入一条垂直线，并且这两列间的距离增大了，所增大的宽度等于这条垂直线的宽度。这是原始 tabular 环境的选项，但 array 宏包对其进行了修正。
新增选项	**说 明**
m{*width*}	文本被排在宽度为 *width* 的段落盒子中，且盒子垂直意义上的中间与同一行中居于其他列的文本的基线对齐。这等同于 \parbox{*width*}。
b{*width*}	等同于 \parbox[b]{*width*}。
>{*text*}	用于列格式选项 l，r，c，p，m 和 b 之前，它将 *text* 插入到对应的列文本之前。
<{*text*}	用于列格式选项 l，r，c，p，m 和 b 之后。它将 *text* 插入到对应的列文本之后。
!{声明}	此选项与 | 相关，指明用声明的内容来取代垂直线；与 @{表达式} 不同，它不删除两列之间的空白。

```
\begin{tabular}{|>{$}l<{$}|l|}\hline
10!^{10!} & a big number\\ \hline
10^{-999} & a small number\\\hline
\end{tabular}
```

$10!^{10!}$	a big number
10^{-999}	a small number

使用选项 !{声明} 还可以改变垂直线的粗细。下面我们先用 TeX 中画垂直直线的命令 {\vrule width 3pt} 来定义一个新命令 \cx，然后把 \cx 作为 !{声明} 中的声明，这样就在表格中画出一条 3pt 粗细的垂直直线，这条垂直线的高度由表格自动确定。如果用命令 \rule 来画，就必须指明确切的高度。

```
\newcommand{\cx}{\vrule width 2pt}
\begin{tabular}{|c!{\cx}c|c|}\hline
A & B & C\\\hline 100 & 10 & 1\\\hline
\end{tabular}
```

A	B	C
100	10	1

从下面三个例子可以看出 p，m，b 这三个列格式的区别。

```
\begin{tabular}{|p{1cm}|p{1cm}|p{1cm}|}
\hline 1 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2 & 3 3 3 3 \\ \hline
\end{tabular}
```

1 1 1 1 1 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2	3 3 3 3

```
\begin{tabular}{|m{1cm}|m{1cm}|m{1cm}|}
\hline 1 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2 & 3 3 3 3 \\ \hline
\end{tabular}
```

1 1 1 1 1 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2	3 3 3 3

```
\begin{tabular}{|b{1cm}|b{1cm}|b{1cm}|}
\hline 1 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2 & 3 3 3 3 \\ \hline
\end{tabular}
```

1 1 1 1 1 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2	3 3 3 3

array 宏包还提供了一个定义新的列格式选项的命令

\newcolumntype{*col*}[*n*]{*decl*}

其中 *col* 表示新列格式选项名，它只能用一个字母表示；*n* 是一个可选项，它是正整数，表示新的列格式选项 *col* 有多少参数；这里的 *decl* 是对 *col* 这个新列格式的定义，它必须是合法的列格式声明。例如，命令 `\newcolumntype{C}{>{$}c<{$}}` 定义了新列格式选项 C，它指明对应的表格列处于居中的数学模式中。命令 `\newcolumntype` 的另外一个不同的但非常方便的用处是它定义的选项不仅可以表示一列，也可以与多列相关。例如，可以先用命令 `\newcolumntype{Z}{clr}` 来定义一个与三列相关的选项 Z，然后用

```
\begin{tabular}{Z} ··· \end{tabular}
```

来建立一个有 3 列的表格，其第 1 列居中，第 2 列左对齐，第 3 列右对齐。

我们知道，用标准 LaTeX 的 tabular 环境建立无框表格时，可以使表格的第 1 行或最后一行的基线与表格外的文本基线对齐，这只要使用 tabular 环境的位置选项 t 或 b 就可以了。然而，在建立一个带有边框的表格时，通常只能使表格的上边线或下边线与表格外的文本基线对齐。下面就是这样一个例子：

```
Tables
\begin{tabular}[t]{|l|}\hline
with some\\ hline\\ commands\\ \hline
\end{tabular}
used.
```

Tables

with some
hline
commands

used.

要解决这个问题，可以利用 array 宏包提供的两条命令

\firsthline 和 \lasthline

其中第一条命令用在 tabular 环境中的最前面，而第二条命令用在 tabular 环境中的最后面。例如：

```
Tables
\begin{tabular}[t]{|l|}\firsthline
with some\\ hline\\ commands\\\lasthline
\end{tabular}
used.
```

Tables

with some
hline
commands

used.

另外 array 宏包还定义了两个长度参数：

\extrarowheight 和 \extratabsurround

其中第一个参数用来增加表格中所有行的高度，同时保持行的深度不变。这在带框表格中是很有用的，因为不用这条命令时，表格中的大写字母有可能碰到上边线。第二个参数用来增加表格环境与上下文之间的距离，它常用于嵌套的表格。例如：

```
\setlength{\extratabsurround}{4pt}
\begin{tabular}{|cc|}\hline
Peter &\begin{tabular}[t]{|cl|}
\firsthline
Month &\multicolumn{1}{c|}{Telephone}
\\\hline Aug & 555443\\ Sep & 221145\\
\lasthline
\end{tabular} \\ \hline
\end{tabular}
```

Peter	Month	Telephone
	Aug	555443
	Sep	221145

4.5 用 booktabs 宏包改变横线的粗细

上节中已阐述了利用 array 宏包可以改变垂直表格线的粗细，那么如何改变水平表格线的粗细呢？当然我们可以用命令 \hline\hline 画水平双线，并重新设置参数命令

\doublerulesep 的值来改变双线之间的距离，从而得到一条较粗的水平线。但这样做不是很方便，也不能适应更多的需要。为此我们介绍 booktabs 宏包，利用它可以很方便地排版不同粗细的水平表格线。

booktabs 宏包提供了下面三条命令

\toprule[*wd*] \midrule[*wd*] \bottomrule[*wd*]

分别用来画表格顶部、中部和底部水平线，其中选项 *wd* 是一个 TeX 长度，表示线的粗细，对这三条命令来说，*wd* 的默认值分别是 0.08em, 0.05em, 0.08em，粗水平线的默认值由参数命令 \heavyrulewidth 定义，而细水平线的默认值由参数命令 \lightrulewidth 定义。

为了在表格中画较短的水平线，booktabs 宏包还提供了下面与命令 \cline 类似的命令

\cmidrule[*wd*](*trim*){*n* − *m*}

可用它画一条从第 *n* 列左边缘起到第 *m* 列右边缘结束，并且粗为 *wd* 的水平线，这里 *wd* 的默认值是 0.03em，并可以用参数命令 \cmidrulewidth 重新设置；*trim* 是一个选项，一般不需要给出，但是当连续使用两条 \cmidrule 命令时，所画的水平线可能衔接不好，此时可以利用这个选项对线条的两端进行修剪，选项的值可以取 r, l, r{*wd*}, l{*wd*} 之一，或者是它们的组合，这里的 *wd* 仍是 TeX 长度，表示在线条的一端可以有多少修剪，若 *wd* 不给出，则可修剪的范围由参数命令 \cmidrulekern 决定，其默认值是 0.5em，比如 lr{0.75em} 表示在线条的左端可以有 0.5em 的修剪，而在右端可以有 0.75em 的修剪。

在命令 \cmidrule 之后紧接着给出命令

\morecmidrules

会引导 LaTeX 在下一行再画一条与前面所画的线等长且对齐的水平线，两条水平线的间距由参数命令 \cmidrulesep 决定，其默认值与 \doublerulesep 的默认值相同。

利用 booktabs 宏包不仅可以改变水平表格线的粗细，还可以改变表格的行距。事实上，命令 \toprule 就会在所画线条的下方增加一个由参数命令 \belowrulesep 决定的垂直空白，同时在上方增加一个由参数命令 \abovetopsep 决定的垂直空白。\abovetopsep 的默认值是 0pt，当表格上面有标题时可以重新定义这个默认值，这样就无需用命令 \vspace 来增加表格标题与表格之间的空白了。

命令 \bottomrule 会在所画线条的上方增加一个由参数命令 \aboverulesep 决定的垂直空白，同时在其下方增加一个有参数命令 \belowbottomsep 决定的垂直空

白。\belowbottomsep 的默认值也是 0pt，当表格下方有内容（如表格脚注、下方表格标题等）可以重新设定这个默认值使之满足需要。

命令 \midrule 会在所画线条的上方增加一个由参数命令 \aboverulesep 决定的垂直空白，同时在其下方增加一个有参数命令 \belowrulesep 决定的垂直空白。

命令 \cmidrule 也会在所画线条的上方增加一个由参数命令 \aboverulesep 决定的垂直空白，除非它紧跟在另一条 \cmidrule 命令之后，在此情形两条水平线之间的空白由 \cmidrulesep 决定。命令 \cmidrule 还会在所画线条的下方增加一个由参数命令 \belowrulesep 决定的垂直空白，除非这条命令后面还是一条同等的画线命令，此时两条水平线之间的空白仍然由 \cmidrulesep 决定。

下面是一条一般性的命令

\specialrule{*wd*}{*abovespace*}{*belowspace*}

其中的参数都是 TeX 长度，它指示画一条粗为 *wd* 且与表格等宽的水平线，线条的上方预留高为 *abovespace* 的空白，而下方预留高为 *belowspace* 的空白。

booktabs 宏包还提供了下面的命令

\addlinespace[*wd*]

用来在表格中增加一个垂直空白，其中选项 *wd* 表示空白的高度，默认值由参数命令 \defaultaddspace 决定。实际上，这条命令等同于 \specialrule{0pt}{0pt}{*wd*}。下面是 booktabs 宏包提供的一个例子：

```
\begin{tabular}{@{}llr@{}}\toprule
\multicolumn{2}{c}{Item}\\\cmidrule(r){1-2}
Animal & Description & Price (\$)\\\midrule
Gnat  & per gram & 13.65 \\
      & each     &  0.01 \\
Gnu   & stuffed  & 92.50 \\
Emu   & stuffed  & 33.33 \\
Armadillo & frozen & 8.99\\ \bottomrule
\end{tabular}
```

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

4.6 自动计算列宽的 tabularx 宏包

David Carlisle 编写的宏包 tabularx 增强了标准 LaTeX 制表环境 tabular* 的功能，它能根据所给表格的总宽度自动计算特定表格列的宽度。这种需要自动计算其宽度的表

格列，必须在相应的列格式中用字母 X 来指明。一旦正确计算出表格列的宽度，列格式就会被转换为 p{某个列宽值}。

tabular* 环境与 tabularx 环境的主要区别在于：

1) tabularx 环境改变列的宽度，而 tabular* 环境改变列与列之间的空白宽度。

2) tabular* 环境与 tabularx 环境都可以嵌套使用。但是 tabularx 环境嵌套使用时，内部表格必须包含在一对花括弧 { } 之中。

例如，下面的表格（用 \begin{tabularx}{10.5cm}{|X|X|X|} 设置）总宽度被设置成 10.5cm，各列的宽度自动计算。

聪明的鱼儿在咬钩前常常徘徊再三	这是因为它们要判断食物是否安全	如果它们认为有危险
它们就不会吃	如果它们判定没有危险	它们就会吞钩
一眼识破诱饵的危险，却又不由自主地去吞钩的	那才正是人的心理而不是鱼的心理	是人的愚蠢而不是鱼的愚蠢

如果将上面表格的设置改为 \begin{tabularx}{\linewidth}{|p{3cm}|X|X|}，则表格的总宽度是行宽，第 1 列列宽为 3cm，其他两列的列宽自动计算。

聪明的鱼儿在咬钩前常常徘徊再三	这是因为它们要判断食物是否安全	如果它们认为有危险
它们就不会吃	如果它们判定没有危险	它们就会吞钩
一眼识破诱饵的危险，却又不由自主地去吞钩的	那才正是人的心理而不是鱼的心理	是人的愚蠢而不是鱼的愚蠢

通常在 tabularx 环境中，所有 X 列都是平均分配宽度的，不过也可以通过设置列格式来改变。例如下面的列格式

```
{>{\setlength{\hsize}{.5\hsize}}X>{\setlength{\hsize}{1.5\hsize}}X}
```

设置了含有两列的表格，其中第 2 列的宽度设置为第 1 列的宽度的 3 倍。但使用这种方法时必须遵循两条规则：第一，所有 X 列的总宽度应是确定值；其次，\multicolumn 命令不能跨越 X 列。

另外，与 array 宏包一样，这里也可以用命令 \newcolumntype 来定义新的列格

式。在 tabularx 宏包中，X 列格式被设置成 p 列格式的形式，它是由参数

\tabularxcolumn

定义的，并且与 \parbox[t] 相关。事实上，X 列格式的原始定义为：

```
\newcommand{\tabularxcolumn}[1]{p{#1}}
```

若要 X 列格式被设置成与 \parbox[c] 相关的 m 列格式形式，则可以使用下面的命令：

```
\renewcommand{\tabularxcolumn}[1]{>{\small}m{#1}}
```

例如：

```
\renewcommand{\tabularxcolumn}%
[1]{>{\small}m{#1}}
\begin{tabularx}{4cm}{|X|X|X|}\hline
1 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2 & 3 3 3 3 \\ \hline
\end{tabularx}
```

1 1 1 1 1 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2	3 3 3 3

4.7 用 multirow 宏包处理跨行表格数据

在第 103 页中我们已举例说明可以用 \raisebox 命令来使表格的文本数据排在两行的中间，下面是另一个小例子：

```
\begin{tabular}{|c|c|c|}\hline
&\multicolumn{2}{c|}{qqq}\\\cline{2-3}
 \raisebox{1.5ex}[0pt]{100} &
          A &  B \\ \hline
2000000 & 10 & 10 \\ \hline
\end{tabular}
```

100	qqq	
	A	B
2000000	10	10

利用 multirow 宏包提供的命令

\multirow{*n*}[*bigstruts*]{*width*}[*vmove*]{文本}

也可以在 tabular 环境中很方便地排版跨行文本数据，其中

n 是正整数表示文本所占的行数。

bigstruts 是可选项，默认值为 0，且只有同时使用 bigstrut 宏包时才有意义，它表示被文本所占的行中使用 \bigstruts 的次数。

width 是文本所在列的宽度。它可以用 * 取代，表示用文本的自然宽度作为列的宽度，但此时 * 不能用花括号括起来，如 \multirow{3}*{text}。

vmove 这个可选项表示文本的垂直位移量，正值往上移动负值往下移动。

文本 是被跨行排版的文本数据。当 *width* 设置为一个确定的值时，文本排在一个段落盒子中，若 *width* 被 * 取代，则文本排在左右盒子中。

还可以用命令参数 \multirowsetup 来改变文本在格子中被排版的方式。这个参数的默认定义就是 \raggedright，即文本在格子中左对齐，因而命令

```
\renewcomand{\multirowsetup}{\centering}
```

就使文本在格子中居中排版。例如：

```
\begin{tabular}{|l|l|}\hline
\multirow{4}{2cm}{Text in Column 1}
& Column g2a\\\cline{2-2}& Column g2b\\
\cline{2-2} & Column g2c \\ \cline{2-2}
& Column g2d \\ \hline
\multirow{4}{2cm}[-1.5ex]%
            {Text in Column 1}
& Column g2a\\\cline{2-2}& Column g2b\\
\cline{2-2} & Column g2c \\ \cline{2-2}
& Column g2d \\ \hline
\end{tabular}
```

Text in Column 1	Column g2a
	Column g2b
	Column g2c
	Column g2d
Text in Columin 1	Column g2a
	Column g2b
	Column g2c
	Column g2d

本小节第一个例子中的小表格也可以按下面的方法排版：

```
\renewcommand{\multirowsetup}%
{\centering}
\begin{tabular}{|c|c|c|}\hline
\multirow{2}*{100} &
\multicolumn{2}{c|}{qqq}\\\cline{2-3}
          &  A &  B \\ \hline
 2000000 & 10 & 10 \\ \hline
\end{tabular}
```

100	qqq	
	A	B
2000000	10	10

4.8 用 dcolumn 宏包使列中的小数点对齐

用标准的 LaTeX 排版由阿拉伯数字及小数点组成的表格时，如果一列中的数字都带有小数点，则可以用 @-{表达式} 这种列格式选项来使小数点对齐，但若一列中又有某些数字为不带小数点的整数时，就很难将这列中的小数点对齐了。Tools 宏包套件中的 dcolumn 宏包就是专为解决此问题的。实际上这个宏包为 tabular 和 array 环境定

义了一个带有三个参数的列格式选项：

D{输入标点}{输出标点}{小数位数}

输入标点 表示源文件中输入的要将之对齐的字符，通常是小数点和逗号。

输出标点 表示排版之后与输入标点对应的已对齐了的字符。它可以与输入标点相同也可以不相同。

小数位数 表示一列中所有标点之后所含字符的最大个数。此数如果为负，则表示标点后面可以含有任意位，此时对齐的标点居于列的中间，因而可能使列显得太宽。

由于这个选项带有三个参数，使用起来可能稍显麻烦，不过结合 array 宏包中的命令 \newcolumntype 可以定义各种简单的选项，例如命令

```
\newcolumntype{d}[1]{D{.}{\cdot}{#1}}
```

定义了带一个参数的选项 d，此时 d{2} 就代表输入小数点，输出居中点且小数点后面最多有两位小数。命令 \newcolumntype{.}{D{.}{.}{-1}} 定义了不带参数的选项“.”，输入标点和输出标点也都是点“.”且小数点后面可以含任意位小数。命令

```
\newcolumntype{,}{D{,}{,}{2}}
```

定义了不带参数的选项“,”，输入标点和输出标点也都是逗号“,”且逗号后面最多可以含两位数。下面的例子使用了上面三个选项。

```
\begin{tabular}{|d{-1}|d{2}|.|,|}
1.23 & 1.23 & 12.5 & 300,2\\
1121.2 & 1121.2 & 861.20 & 674,29\\
184 & 184 & 10 & 69 \\
.4 & .4 & .4 & ,4
\end{tabular}
```

1·23	1·23	12.5	300,2
1121·2	1121·2	861.20	674,29
184	184	10	69
·4	·4	.4	,4

4.9 用 hhline 宏包调整水平线与垂直线的交叉点

在带框的表格中总会有水平直线与垂直直线相交的情况。两条水平直线与两条垂直直线相交时，在交点处会出现各种形态。这种形态在标准的 LaTeX 表格环境中是不可改变的。hhline 宏包提供了一条类似于 \hline 的画线命令，即

\hhline{声明}

用它可以很好地处理水平直线与垂直直线的相交形态，其中声明里可以包含如下一些顺序排列的选项：

= 表示与列等宽的两条水平直线。
- 表示与列等宽的一条水平直线。
~ 表示没有水平直线，只有一个与列等宽的空白。
| 表示一条垂直直线穿过两条水平直线。
: 表示一条垂直直线被两条水平直线切断。
表示两条水平直线段与两条垂直直线相交。
t 表示两条水平直线段中上边的线段。
b 表示两条水平直线段中下边的线段。
* 表示给定的参数选项重复出现。如 *{3}{===#} 等价于 ===#===#===#。

如果在声明中用 || 或 :: 来排版双垂线，那么由 \hhline 产生的水平直线会被切断。要得到“一条水平线被双垂线窜过”的效果，可以根据情况使用 # 或者省略垂线的声明。

选项 t 和 b 可以用于两条垂直直线中间，例如 |tb| 产生的交点形态与 # 一样，但并没有那么有效。它的主要作用是用来构造 |t:（左上角）和 :t|（右上角）以及 |b:（左下角）和 :b|（右下角）。

如果要用命令 \hhline 画一条水平直线，则在选项中应该只包含 -, ~ 和 |。下面是 hhline 宏包中提供的一个例子：

```
\begin{tabular}{||cc||c|c||}
\hhline{|t:==:t:==:t|} a & b & c & d\\
\hhline{|:==:|~|~||} 1 & 2
&3& 4\\\hhline{#==#~|=#} i& j& k& l\\
\hhline{||--||--||} w
& x & y & z\\ \hhline{|b:==:b:==:b|}
\end{tabular}
```

a	b	c	d
1	2	3	4
i	j	k	l
w	x	y	z

注意，在标准 LaTeX 中，命令 \hline 的定义只用了一个 TeX 的底层画线命令 \hrule，而 hhline 宏包中的命令 \hhline 用了许多这种画小线段的命令。在排版时，TeX 系统会将这些信息准确地记录在 dvi 文件中，但是用来显示和打印 dvi 文件的驱动程序也许不能很好地将这些小线段排列整齐。此时可以将参数 \arrayrulewidth 的值设置得更大些来改善这种情况。

4.10 用 longtable 宏包创建跨页表格

Tools 宏包套件中的 longtable 宏包定义了一个新的创建表格的 longtable 环境。这个环境具有 tabular 环境的大多数特点，但由它创建的表格可以跨页排版。此环境的

使用语法如下：

```
\begin{longtable}[位置]{列格式}  表格行  \end{longtable}
```

其中位置是可选项，可以省略也可选择 l, c, r 三个选项之一，分别表示整个表格在水平方向上居左、居中和居右。列格式和表格行的使用方法与 tabular 环境中的列格式和表格行的使用方法大致相同。

虽然 TeX 系统在必要时会将由 longtable 环境建立的表格分成多页排版，但它不会把表格从某行的中间分开，这对那些包含在段落盒子中的表格文本是很重要的。我们也可以在表格行的前面使用 \newpage, \pagebreak, \nopagebreak 等命令来控制表格是否从此处换页。另外还可以在表格中使用 \footnote, \footnotemark 和 \footnotetext 这些排印脚注的命令，但 \footnote 不能用于表格头和表格尾中。表 4.2 中列出了其他所有控制 longtable 环境的命令、参数和选项。

表 4.2 控制 longtable 环境的命令、参数和选项

参数及默认值	说 明
\LTleft (\fill)	用于环境前，设置表格与页面左边缘之间的距离。
\LTright (\fill)	用于环境前，设置表格与页面右边缘之间的距离。
\LTpre (\bigskipamount)	用于环境前，设置表格与上面文本之间的距离。
\LTpost (\bigskipamount)	用于环境前，设置表格与下面文本之间的距离。
\LTchunksize (20)	用于环境前，设置表格块包含的行数。
\LTcapwidth (4in)	用于环境前，设置包含表格标题的段落盒子的宽度。
位置选项	**说 明**
省略	此时表格位置由 \LTleft 和 \LTright 确定。
l	此时表格居左排印。
c	此时表格居中排印。
r	此时表格居右排印。
表格换行命令	**说 明**
\\	通用的换行符。
\\[*dim*]	换行后附加一个长度为 *dim* 的垂直空白。
*	换行，但不允许换行后另起新页。
\tabularnewline	当行中使用了命令 \raggedright 时，应用此换行命令。

续表

\kill	行被删除，用于确定列宽。
\endhead	以此结尾的行是表格头，排在每页表格的顶端。
\endfirsthead	以此结尾的行是首页表格头，排在表格首页的顶端。
\endfoot	以此结尾的行是表格尾，排在每页表格的尾部。
\endlastfoot	以此结尾的行是末页表格尾，排在表格最后一页的尾部。

表格标题命令	说　明
\caption{标题}	标题形式为“Table ?: 标题”，并且标题将被写入表格目录中。
\caption[短标题]{标题}	标题形式为“Table ?: 标题”，短标题将被写入表格目录中。
\caption[]{标题}	标题形式为“Table ?: 标题”，但表格目录中不作记录。
\caption*{标题}	标题形式为“标题”，但表格目录中不作记录。

为了将长表格分成多页来排版，有必要把表格分成若干小块，这样才能不使表格中所有内容一次同时记录到内存中。在默认设置下，longtable 环境将表格小块包含的行数设置为 20，但这个数可以用命令

```
\setcounter{LTchunksize}{{行数}
```

来重新设置。较大的 LTchunksize 使 TeX 有较快的编译速度，但也占用更多的内存。如果有必要可将 LTchunksize 的值设置成 1，此时表格所占的内存几乎可以忽略。然而当表格中有表格头或表格尾时，LTchunksize 的值不应该比表格头或表格尾包含的行数更小。

所谓表格头就是出现在表格每一页顶部的一行或多行内容。表格头以通常表格行的方式输入，但必须用 \endhead 结尾，而不是用 \\。如果第 1 页的表格头与其他页的表格头不同，那么第 1 页的表格头应用 \endfirsthead 结尾。表格尾就是出现在表格每一页底部的一行或多行内容。表格尾的输入行以 \endfoot 结尾，而最后一页的表格尾的输入行以 \endlastfoot 结尾。

表格的标题可以用命令

```
\caption[短标题]{标题}
```

来输入。上面这个标题命令以及它的变形 \caption* 实际上等价于一条特殊的如下形式的命令：

```
\multicolumn{n}{c}{\parbox{\LTcapwidth}{···}}
```

其中 n 是表格的列数。标题的宽度可以用长度参数 \LTcapwidth 来控制，也就是说可以在表格环境之前或者在导言部分使用命令 \setlength{\LTcapwidth}{*width*} 将标题的宽度设置为 *width*。

若表格的每一页都有标题，而首页的标题又与其他表格页的标题不同，此时可将命令 \caption{首页标题} 写在首页标题头的输入行中，即 \endfirsthead 的前面，而将命令 \caption[]{标题} 写在标题头的输入行中，即 \endfhead 的前面。这样只有首页标题记录到表格目录中。

在默认设置下，由 longtable 环境建立的表格是居中排版的，因为此时 \LTleft 和 \LTright 的值都是弹性长度 \fill。当然可以将这两个长度参数的值设置为其他的数，但其中一个应为弹性长度，除非在列格式选项中已利用 \extracolsep 命令设置了一个弹性长度。例如，可以按照下面的方法

```
\setlength{\LTleft}{0pt} \setlength{\LTright}{0pt}
\begin{longtable}{l@{\extracolsep{\fill}}l}
.....................................
\end{longtable}
```

建立一个与页面等宽的表格。图 4.1 是一个由 longtable 环境创建的长表格的例子。

这个有 4 页的长表格在源文件中的输入形式如下：

```
\begin{longtable}{|llrr|}
\hline
\multicolumn{4}{|c|}{\bfseries 国际电话通达国家、代码及资费标准}\\\hline
国家 & 中文国名 & 代码 & 资费/每分钟 \\ \hline\hline\endfirsthead
\multicolumn{4}{r}{续表}\\ \hline
国家 & 中文国名 & 代码 & 资费/每分钟 \\ \hline\hline\endhead
Afghanistan & 阿富汗       & 93   & 23.00 \\ \hline
Albania     & 阿尔巴尼亚   & 355  & 20.70 \\ \hline
Algeria     & 阿尔及利亚   & 213  & 27.60 \\ \hline
Andorra     & 安道尔       & 376  & 20.70 \\ \hline
Angola      & 安哥拉       & 244  & 27.60 \\ \hline
Argentina   & 阿根廷       & 54   & 27.60 \\ \hline
Armenia     & 亚美尼亚     & 374  & 18.90 \\ \hline
Ascension   & 阿森松（英） & 247  & 27.60 \\ \hline
Australia   & 澳大利亚     & 61   & 18.40 \\ \hline
Austria     & 奥地利       & 43   & 20.70 \\ \hline
Azerbaijan  & 阿塞拜疆     & 994  & 18.90 \\ \hline
Bahrain     & 巴林         & 973  & 27.60 \\ \hline
Bangladesh  & 孟加拉国     & 880  & 17.20 \\ \hline
Belarus     & 白俄罗斯     & 375  & 18.90 \\ \hline
```

国际电话通达国家、代码及资费标准

国家	中文国名	代码	资费/每分钟
Afghanistan	阿富汗	93	23.00
Albania	阿尔巴尼亚	355	20.70
Algeria	阿尔及利亚	213	27.60
Andorra	安道尔	376	20.70
Angola	安哥拉	244	27.60
Argentina	阿根廷	54	27.60
Armenia	亚美尼亚	374	18.90
Ascension	阿森松（英）	247	27.60
Australia	澳大利亚	61	18.40
Austria	奥地利	43	20.70
Azerbaijan	阿塞拜疆	994	18.90
Bahrain	巴林	973	27.60
Bangladesh	孟加拉国	880	17.20
Belarus	白俄罗斯	375	18.90
Belgium	比利时	32	20.70
Bhutan	不丹	975	19.50

续表

国家	中文国名	代码	资费/每分钟
Bolivia	玻利维亚	591	27.60
Botswana	博茨瓦纳	267	27.60
Bosnia Herzegovina	波斯尼亚黑塞哥维纳	387	20.70
Brazil	巴西	55	27.60
Brunei	文莱	673	20.70
Bulgaria	保加利亚	359	20.70
Burkina-faso	布基纳法索	226	27.60
Burundi	布隆迪	257	27.60
Cameroon	喀麦隆	237	27.60
Canada	加拿大	1	18.40
Canaries Is	加拿利群岛	34	20.70
Cape Verde	佛得角	238	27.60
Central Africa	中非	236	27.60
Chad	乍得	235	27.60
Congo	刚果	242	27.60

续表

国家	中文国名	代码	资费/每分钟
Costa Rica	哥斯达黎加	506	27.60
Croatia	克罗地亚	384 385	26.70
Cuba	古巴	53	27.60
Cyprus	塞浦路斯	357	20.70
Czech	捷克	420	20.70
Denmark	丹麦	45	20.70
Djibouti	吉布提	253	27.60
Ecuador	厄瓜多尔	593	27.60
Egypt	埃及	20	20.70
El Salvador	萨尔瓦多	503	27.60
Eritrea	厄立特里亚	291	27.60
Estonia	爱沙尼亚	372	18.90
Finland	芬兰	358	20.70
France	法国	53	20.70
Gambia	冈比亚	220	27.60

续表

国家	中文国名	代码	资费/每分钟
Georgia	格鲁吉亚	995	18.90
Germany	德国	49	20.70
Greece	希腊	30	20.70
Guatemala	危地马拉	502	27.60
Guinea	几内亚	224	27.60
Haiti	海地	509	27.60
Hungary	匈牙利	36	20.70
Iceland	冰岛	354	20.70
India	印度	91	19.50
Indonesia	印度尼西亚	62	18.40
Iran	伊朗	98	20.70
Iraq	伊拉克	964	20.70
Ireland	爱尔兰	353	20.70
Israel	以色列	972	20.70
Italy	意大利	39	20.70
Japan	日本	81	12.70

图 4.1 longtable 表格示例

```
Bhutan      & 不丹          & 975   & 19.50 \\ \hline
Bolivia     & 玻利维亚       & 591   & 27.60 \\ \hline
Botswana    & 博茨瓦纳       & 267   & 27.60 \\ \hline
Bosnia      & 波斯尼亚       &       &       \\
Herzegovina & 黑塞哥维纳 &\rb{387}& \rb{20.70} \\\hline
Brazil      & 巴西           & 55    & 27.60 \\ \hline
.....................
Italy       & 意大利         & 39    & 20.70 \\ \hline
Japan       & 日本           & 81    & 12.70 \\ \hline
\end{longtable}
```

其中命令 \rb 的定义见第 103 页。

第5章 数学公式

标准 LaTeX 已经具有很强的排版数学公式的能力。然而当经常性地使用某些复杂的数学公式和数学结构时，用户也有必要自己定义一些新命令和环境来简化源文件的编辑过程。这对于许多 LaTeX 用户来说并不是一件很容易的事，而且每个用户都有自己的定义方法，使得源文件不便阅读。为了规范和简化数学公式的排版，1982 年美国数学会（AMS）在 TeX 的基础上开发了 AMSTeX，其中定义了一套排版数学公式的命令。例如用来排版矩阵的命令 \matrix 以及在数学公式中插入文字的命令 \text 等，这使排版数学文稿变得容易了。但是 AMSTeX 不像 LaTeX 那样提供了用于排版文稿的一些重要功能，如自动处理标号的功能、用于排版索引和对参考文献的交叉引用等功能。这些功能对作者很重要，当 LaTeX 在 1985 年左右被迅速推广时，许多数学杂志也开始要求作者提供手稿的 LaTeX 源文件。在广大 LaTeX 用户的要求下，美国数学会于 1987 年开始将 AMSTeX 移植到 LaTeX，由此便产生了 AMSLaTeX。初版 AMSLaTeX 于 1990 年公开，目前最新的版本是 AMSLaTeX 2.0[1]，它是 LaTeX2e 的一个宏包套件。大多数 TeX 系统，如 MiKTeX、teTeX 等都已安装了 AMSLaTeX。本章主要阐述如何使用 AMSLaTeX 2.0 来排版数学公式。

5.1 AMSLaTeX 宏包套件及 AMSFonts

AMSLaTeX 宏包套件包含一组独立的宏包，其中最常用的一个宏包是 amsmath，在源文件的导言部分输入命令 \usepackage{amsmath} 之后就可以使用这个宏包了。其他的宏包也都可以通过命令 \usepackage 加载到源文件中。要注意加载 amsmath 时，amsbsy, amsopn 和 amstext 这三个宏包也自动被加载了。下面列出了 AMSLaTeX 2.0 宏包套件包含的宏包。

amsmath　定义了各种显示多行公式的环境和一系列排版数学公式的命令。

amsbsy　定义了排版黑体数学符号的命令 \boldsymbol 和 \pmb (poor man's bold)。

amstext　定义了命令 \text 用来在数学公式中插入纯文本文字。

amsopn　提供了命令 \DeclareMathOperator，用来定义类似于 \sim，\lim 等新的算符和函数名称。

amscd　定义了 CD 环境和一些简化交换图表的排版命令，但不支持对角箭头。

amsthm　定义了 proof 环境和扩充的 \newtheorem 命令，用来排版定理和证明。

[1] AMSLaTeX 2.0 中有两个名为 diffs-?.txt 的文件，其中列出了各个版本之间的不同。

amsxtra 提供了一些不太常用的命令，例如用来排版带各种括号的分式的命令 \fracwithdelims，和排版带重音的数学符号命令 \accentedsymbol。

upref 使得用 \ref 来排版交叉引用标号时，总是使用直立的罗马字体。

调用 amsmath 宏包时，可以根据不同情况使用下面一些选项，它们关系到数学算符的上、下限和公式标号的位置。

centertags 这是默认选项，表示用 split 环境排版多行数学公式时，公式标号放在垂直位置的中间。

tbtags 表示用 split 环境排版多行数学公式时，如果公式标号放在右边，则公式标号与公式的最后一行（bottom）对齐。若公式标号放在左边，则公式标号与公式的第 1 行（top）对齐。

intlimits 这是默认选项，表示在独立显示的数学公式中积分符号的上限和下限分别放在积分符号的顶部和底部。

nointlimits 若使用这个选项，则无论文本中的还是独立显示的数学公式，积分符号的上限和下限都放在旁边。

namelimits 这是默认选项，表示按照数学习惯在独立显示的（displaystyle）数学公式中，诸如 lim, det, inf, max, min 这些算符的下标都放在符号的下方。

nonamelimits 与 namelimits 相反，无论是文本中的数学公式还是独立显示的数学公式，下标总是放在旁边。

sumlimits 这是默认选项，表示在独立显示的（displaystyle）数学公式中，求和符号的上标和下标分别放在符号的上方和下方。这个选项对其他的同类符号如 $\prod$, $\coprod$, $\bigotimes$, $\bigoplus$ 等也起作用，但不包含积分号。

nosumlimits 即使在独立显示的（displaystyle）数学公式中，求和符号以及同类的其他符号的上标和下标也都放在符号的旁边。

使用 AMSTeX 时，由 AMS 构造的数学符号所用的字体（AMSFonts）也被自动加载了。但是在 AMSLaTeX 中，为了升级的方便，AMSFonts 被分离出来成为几个独立的宏包，它们包含下面几个宏包，其中最常用的是 amssymb。

amsfonts 它定义了命令 \mathfrak 和 \mathbb, 并且设置了数学公式中使用的一些字体，如

- msam extra math sybmol A.
- msbm extra math sybmol B 和 black-board bold.
- eufm Euler Fraktur.
- cmmib bold math italic 和 bold lowercase Greek.

cmbsy bold math symbols 和 bold script.

amssymb 定义了 AMS 提供的数学符号的命令。使用这个宏包时，amsfonts 也被同时加载了。

eufrak 这个宏包设置了 Fraktur 字体。

eucal 当使用此宏包时，命令 \mathcal 调用 Euler script 字体，而不是通常的 Computer Modern script 字体。

5.2 数学公式中的符号与字体

5.2.1 数学符号

在排版数学公式时，不可避免地会遇到各种各样的数学符号。虽然标准 LaTeX 已提供了很多数学符号，但在科技发展的今天，这些符号还不够使用，因此 AMS 提供许多额外的数学符号。表 5.1 到表 5.11 列出了标准 LaTeX 中所提供的大多数数学符号。利用命令 \not 可以在数学符号上面加一个斜杠来产生像“不等于”、“不属于”这类具有否定意义的符号。例如：

```
$u\not<v$ or $a\not\in\mathbf{A}$
```

$u \not< v$ or $a \notin \mathbf{A}$

从 125 页中的表 5.12 到 128 页中的表 5.19 列出了 AMSFonts 中的额外数学符号。使用 amssymb 宏包，就可以把这些数学符号一次加载到系统中。然而如果用户只想使用其中个别符号，或者由于内存的原因 amssymb 不能将所有这些数学符号一次加载到系统中，那么也可以只加载 amsfonts 宏包，并且利用 \DeclareMathSymbol 命令来自己定义一些必要的数学符号。

表 5.1 定界符号（在 LaTeX 中有效）

符号及其命令		符号及其命令		符号及其命令		符号及其命令	
$\uparrow$	\uparrow	$\Uparrow$	\Uparrow	$\downarrow$	\downarrow	$\Downarrow$	\Downarrow
$\{$	\{	$\}$	\}	$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil	$\rceil$	\rceil
$\langle$	\langle	$\rangle$	\rangle	/	/	\	\backslash
$\vert$	|	$\Vert$	\\|				

表 5.2 希腊字母（在 LaTeX 中有效）

符号及其命令		符号及其命令		符号及其命令		符号及其命令		符号及其命令	
α	\alpha	β	\beta	γ	\gamma	δ	\delta	ϵ	\epsilon
ε	\varepsilon	ζ	\zeta	η	\eta	θ	\theta	ϑ	\vartheta
ι	\iota	κ	\kappa	λ	\lambda	μ	\mu	ν	\nu
ξ	\xi	o	o	π	\pi	ϖ	\varpi	ρ	\rho
ϱ	\varrho	σ	\sigma	ς	\varsigma	τ	\tau	υ	\upsilon
ϕ	\phi	φ	\varphi	χ	\chi	ψ	\psi	ω	\omega
Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda	Ξ	\Xi
Π	\Pi	Σ	\Sigma	Υ	\Upsilon	Φ	\Phi	Ψ	\Psi
Ω	\Omega								

表 5.3 箭头符号（在 LaTeX 中有效）

符号及其命令		符号及其命令		符号及其命令	
$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow

表 5.4 可变尺寸符号（在 LaTeX 中有效）

符号及其命令		符号及其命令		符号及其命令		符号及其命令		符号及其命令	
$\sum$	\sum	$\prod$	\prod	$\coprod$	\coprod	$\int$	\int	$\oint$	\oint
$\bigcap$	\bigcap	$\bigcup$	\bigcup	$\bigsqcup$	\bigsqcup	$\bigvee$	\bigvee	$\bigwedge$	\bigwedge
$\bigodot$	\bigodot	$\bigotimes$	\bigotimes	$\bigoplus$	\bigoplus	$\biguplus$	\biguplus		

表 5.5 大定界符号（在 LaTeX 中有效）

符号及其命令	符号及其命令	符号及其命令	符号及其命令
⎱ \rmoustache	⎰ \lmoustache	) \rgroup	(\lgroup
⎪ \arrowvert	‖ \Arrowvert	⎪ \bracevert	

表 5.6 常见函数的命令（在 LaTeX 中有效）

函数及命令		函数及命令		函数及命令		函数及命令	
arccos	\arccos	cos	\cos	min	\min	ker	\ker
arcsin	\arcsin	sin	\sin	max	\max	gcd	\gcd
arctan	\arctan	tan	\tan	deg	\deg	hom	\hom
lim inf	\liminf	coth	\coth	lim	\lim	inf	\inf
lim sup	\limsup	sec	\sec	dim	\dim	sup	\sup
cosh	\cosh	csc	\csc	log	\log	lg	\lg
sinh	\sinh	cot	\cot	exp	\exp	ln	\ln
tanh	\tanh	arg	\arg	det	\det	Pr	\Pr

表 5.7 二元算符（在 LaTeX 中有效）

符号及其命令	符号及其命令	符号及其命令	符号及其命令
± \pm	∩ \cap	◇ \diamond	⊕ \oplus
∓ \mp	∪ \cup	△ \bigtriangleup	⊖ \ominus
× \times	⊎ \uplus	▽ \bigtriangledown	⊗ \otimes
÷ \div	⊓ \sqcap	◁ \triangleleft	⊘ \oslash
* \ast	⊔ \sqcup	▷ \triangleright	⊙ \odot
⋆ \star	∨ \vee	◁ \lhd[a]	◯ \bigcirc
∘ \circ	∧ \wedge	▷ \rhd[a]	† \dagger
• \bullet	\ \setminus	⊴ \unlhd[a]	‡ \ddagger
· \cdot	≀ \wr	⊵ \unrhd[a]	⨿ \amalg

[a] 在 NFSS 中无定义。需要 latexsym 或者 amssymb 宏包。

表 5.8 关系符号（在 LaTeX 中有效）

符号及其命令	符号及其命令	符号及其命令	符号及其命令	符号及其命令
$\leq$ \leq, \le	$\geq$ \geq, \ge	$\equiv$ \equiv	$\models$ \models	$\prec$ \prec
$\succ$ \succ	$\sim$ \sim	$\perp$ \perp	$\preceq$ \preceq	$\succeq$ \succeq
$\simeq$ \simeq	$\mid$ \mid	$\ll$ \ll	$\gg$ \gg	$\asymp$ \asymp
$\parallel$ \parallel	$\subset$ \subset	$\supset$ \supset	$\approx$ \approx	$\sqsubseteq$ \sqsubseteq
$\subseteq$ \subseteq	$\supseteq$ \supseteq	$\cong$ \cong	$\bowtie$ \Join	$\sqsubset$ \sqsubset
$\sqsupset$ \sqsupset	$\neq$ \neq	$\smile$ \smile	$\bowtie$ \bowtie	$\sqsupseteq$ \sqsupseteq
$\doteq$ \doteq	$\frown$ \frown	$\in$ \in	$\ni$ \ni	$\propto$ \propto
$=$ =	$\vdash$ \vdash	$\dashv$ \dashv	$<$ <	$>$ >

表 5.9 数学符号的重音（在 LaTeX 中有效）

符号及其命令	符号及其命令	符号及其命令	符号及其命令	符号及其命令
$\hat{a}$ \hat {a}	$\acute{a}$ \acute {a}	$\bar{a}$ \bar {a}	$\dot{a}$ \dot {a}	$\breve{a}$ \breve {a}
$\check{a}$ \check {a}	$\grave{a}$ \grave {a}	$\vec{a}$ \vec {a}	$\ddot{a}$ \ddot {a}	$\tilde{a}$ \tilde {a}

表 5.10 其他各种符号（在 LaTeX 中有效）

符号及其命令	符号及其命令	符号及其命令	符号及其命令
$\ldots$ \ldots	$\cdots$ \cdots	$\vdots$ \vdots	$\ddots$ \ddots
$\prime$ \prime	$\forall$ \forall	∞ \infty	$\hbar$ \hbar
$\exists$ \exists	∇ \nabla	$\surd$ \surd	$\Box$ \Box[a]
$\Diamond$ \Diamond[a]	$\imath$ \imath	$\jmath$ \jmath	ℓ \ell
$\top$ \top	$\flat$ \flat	$\natural$ \natural	$\sharp$ \sharp
$\bot$ \bot	$\clubsuit$ \clubsuit	$\diamondsuit$ \diamondsuit	$\heartsuit$ \heartsuit
$\mho$ \mho[a]	$\Re$ \Re	$\Im$ \Im	$\angle$ \angle
$\aleph$ \aleph	$\emptyset$ \emptyset	$\triangle$ \triangle	$\neg$ \neq
$\wp$ \wp	$\spadesuit$ \spadesuit	∂ \partial	

[a] 在 NFSS 中无定义。需要 latexsym 或者 amssymb 宏包。

表 5.11 LaTeX 中的数学构造

符号及其命令		符号及其命令	
$\widetilde{abc}$	\widetilde {abc}	$\widehat{abc}$	\widehat {abc}
$\overleftarrow{abc}$	\overleftarrow {abc}	$\overrightarrow{abc}$	\overrightarrow {abc}
$\overline{abc}$	\overline {abc}	$\underline{abc}$	\underline {abc}
$\overbrace{abc}$	\overbrace {abc}	$\underbrace{abc}$	\underline {abc}
$\sqrt{abc}$	\sqrt {abc}	$\sqrt[n]{abc}$	\sqrt [n]{abc}
f'	f'	$\frac{abc}{xyz}$	\frac {abc}{xyz}

表 5.12 AMS 二元否定关系符（需要 amssymb 宏包）

符号及其命令		符号及其命令		符号及其命令	
$\nless$	\nless	$\nleq$	\nleq	$\nleqslant$	\nleqslant
$\nleqq$	\nleqq	$\lneq$	\lneq	$\lneqq$	\lneqq
$\lvertneqq$	\lvertneqq	$\lnsim$	\lnsim	$\lnapprox$	\lnapprox
$\nprec$	\nprec	$\npreceq$	\npreceq	$\precnsim$	\precnsim
$\precnapprox$	\precnapprox	$\nsim$	\nsim	$\nshortmid$	\nshortmid
$\nmid$	\nmid	$\nvdash$	\nvdash	$\nvDash$	\nvDash
$\ntriangleleft$	\ntriangleleft	$\ntrianglelefteq$	\ntrianglelefteq	$\nsubseteq$	\nsubseteq
$\subsetneq$	\subsetneq	$\varsubsetneq$	\varsubsetneq	$\subsetneqq$	\subsetneqq
$\varsubsetneqq$	\varsubsetneqq	$\ngtr$	\ngtr	$\ngeq$	\ngeq
$\ngeqslant$	\ngeqslant	$\ngeqq$	\ngeqq	$\gneq$	\gneq
$\gneqq$	\gneqq	$\gvertneqq$	\gvertneqq	$\gnsim$	\gnsim
$\gnapprox$	\gnapprox	$\nsucc$	\nsucc	$\nsucceq$	\nsucceq
$\succnsim$	\succnsim	$\succnapprox$	\succnapprox	$\ncong$	\ncong
$\nshortparallel$	\nshortparallel	$\nparallel$	\nparallel	$\nvDash$	\nvDash
$\nVDash$	\nVDash	$\ntriangleright$	\ntriangleright	$\ntrianglerighteq$	\ntrianglerighteq
$\nsupseteq$	\nsupseteq	$\nsupseteqq$	\nsupseteqq	$\supsetneq$	\supsetneq
$\varsupsetneq$	\varsupsetneq	$\supsetneqq$	\supsetneqq	$\varsupsetneqq$	\varsupsetneqq

表 5.13 AMS 否定箭头（需要 amssymb 宏包）

符号及其命令		符号及其命令		符号及其命令	
$\nleftarrow$	\nleftarrow	$\nrightarrow$	\nrightarrow	$\nLeftarrow$	\nLeftarrow
$\nRightarrow$	\nRightarrow	$\nleftrightarrow$	\nleftrightarrow	$\nLefrighttarrow$	\nLefrighttarrow

表 5.14 AMS 二元关系符（需要 amssymb 宏包）

符号及其命令		符号及其命令		符号及其命令	
$\leqq$	\leqq	$\leqslant$	\leqslant	$\eqslantless$	\eqslantless
$\lesssim$	\lesssim	$\lessapprox$	\lessapprox	$\approxeq$	\approxeq
$\lessdot$	\lessdot	$\lll$	\lll, \llless	$\lessgtr$	\lessgtr
$\lesseqgtr$	\lesseqgtr	$\lesseqqgtr$	\lesseqqgtr	$\doteqdot$	\doteqdot, \Doteq
$\risingdotseq$	\risingdotseq	$\fallingdotseq$	\fallingdotseq	$\backsim$	\backsim
$\backsimeq$	\backsimeq	$\subseteqq$	\subseteqq	$\Subset$	\Subset
$\sqsubset$	\sqsubset	$\preccurlyeq$	\preccurlyeq	$\curlyeqprec$	\curlyeqprec
$\precsim$	\precsim	$\precapprox$	\precapprox	$\vartriangleleft$	\vartriangleleft
$\trianglelefteq$	\trianglelefteq	$\vDash$	\vDash	$\Vvdash$	\Vvdash
$\smallsmile$	\smallsmile	$\smallfrown$	\smallfrown	$\bumpeq$	\bumpeq
$\Bumpeq$	\Bumpeq	$\geqq$	\geqq	$\geqslant$	\geqslant
$\eqslantgtr$	\eqslantgtr	$\gtrsim$	\gtrsim	$\gtrapprox$	\gtrapprox
$\gtrdot$	\gtrdot	$\ggg$	\ggg, \gggtr	$\gtrless$	\gtrless
$\gtreqless$	\gtreqless	$\gtreqqless$	\gtreqqless	$\eqcirc$	\eqcirc
$\circeq$	\circeq	$\triangleq$	\triangleq	$\thicksim$	\thicksim
$\thickapprox$	\thickapprox	$\supseteqq$	\supseteqq	$\Supset$	\Supset
$\sqsupset$	\sqsupset	$\succcurlyeq$	\succcurlyeq	$\curlyeqsucc$	\curlyeqsucc
$\succsim$	\succsim	$\succapprox$	\succapprox	$\vartriangleright$	\vartriangleright
$\trianglerighteq$	\trianglerighteq	$\Vdash$	\Vdash	$\shortmid$	\shortmid
$\shortparallel$	\shortparallel	$\between$	\between	$\pitchfork$	\pitchfork
$\varpropto$	\varpropto	$\blacktriangleleft$	\blacktriangleleft	$\therefore$	\therefore
$\backepsilon$	\backepsilon	$\blacktriangleright$	\blacktriangleright	$\because$	\because

表 5.15 AMS 定界符号（需要 amssymb 宏包）

符号及其命令	符号及其命令	符号及其命令	符号及其命令
⌜ \ulcorner	⌝ \urcorner	⌞ \llcorner	⌟ \lrcorner

表 5.16 AMS 二元算符（需要 amssymb 宏包）

符号及其命令	符号及其命令	符号及其命令
∔ \dotplus	⧵ \smallsetminus	⋒ \Cap,\doublecap
⋓ \Cup,\doublecup	⩒ \barwedge	⩡ \veebar
⩣ \doublebarwedge	⊟ \boxminus	⊠ \boxtimes
⊡ \boxdot	⊞ \boxplus	⋇ \divideontimes
⋉ \ltimes	⋊ \rtimes	⋋ \leftthreetimes
⋌ \rightthreetimes	⋏ \curlywedge	⋎ \curlyvee
⊝ \circleddash	⊛ \circledast	⊚ \circledcirc
⋅ \centerdot	⊺ \intercal	

表 5.17 AMS 箭头符号（需要 amssymb 宏包）

符号及其命令	符号及其命令	符号及其命令
⇛ \Rrightarrow	⇝ \rightsquigarrow	⬱ \leftleftarrows
⇆ \leftrightarrows	⇚ \Lleftarrow	↞ \twoheadleftarrow
↢ \leftarrowtail	↫ \looparrowleft	⇋ \leftrightharpoons
↶ \curvearrowleft	↺ \circlearrowleft	↰ \Lsh
⇈ \upuparrows	↿ \upharpoonleft	⇃ \downharpoonleft
⊸ \multimap	↭ \leftrightsquigarrow	⇄ \rightleftarrows
⇉ \rightrightarrows	↠ \twoheadrightarrow	↣ \rightarrowtail
↬ \looparrowright	⇌ \rightleftharpoons	↷ \curvearrowright
↻ \circlearrowright	↱ \Rsh	⇊ \downdownarrows
⇂ \downharpoonright	↾ \upharpoonright	⇢ \dashrightarrow
⇠ \dashleftarrow		

表 5.18 AMS Greek 和 Hebrew （需要 amssymb 宏包）

符号及其命令	符号及其命令	符号及其命令	符号及其命令	符号及其命令
F \digamma	$\varkappa$ \varkappa	$\beth$ \beth	$\daleth$ \daleth	$\gimel$ \gimel

表 5.19 AMS 杂项符号（需要 amssymb 宏包）

符号及其命令	符号及其命令	符号及其命令
$\hbar$ \hbar	$\hslash$ \hslash	$\vartriangle$ \vartriangle
$\triangledown$ \triangledown	$\square$ \square	$\lozenge$ \lozenge
$\circledS$ \circledS	$\angle$ \angle	$\measuredangle$ \measuredangle
$\nexists$ \nexists	$\mho$ \mho	$\Finv$ \Finv
$\Game$ \Game	$\Bbbk$ \Bbbk	$\backprime$ \backprime
$\varnothing$ \varnothing	$\blacktriangle$ \blacktriangle	$\blacktriangledown$ \blacktriangledown
$\blacksquare$ \blacksquare	$\blacklozenge$ \blacklozenge	$\bigstar$ \bigstar
$\sphericalangle$ \sphericalangle	$\complement$ \complement	$\eth$ \eth
$\diagup$ \diagup	$\diagdown$ \diagdown	

5.2.2 数学公式中的字体命令

在数学公式中也可以有限度地改变字体，比如可以将数学符号改为黑体，将公式中的大写英文字母改为双轮廓的字体或者花写字体等。表 5.20 中列出了 AMSLaTeX 提供的数学公式中的字体命令，并且对每一种情形都举了一个例子来说明。另外，也可以使用表 3.6 中的数学字体命令。

在 amsmath 宏包中，命令 \boldsymbol 可以用来排版黑体的数学符号和希腊字母（英文字母除外）。数学公式中直立的黑体英文字母可以用命令 \mathbf 来排印。例如，命令 \boldsymbol{\infty}, \boldsymbol{\pi}, \boldsymbol{+}, \mathbf{A} 分别排出 $\boldsymbol{\infty}$, $\boldsymbol{\pi}$, $\boldsymbol{+}$ 和 $\mathbf{A}$。

为了方便而快速的编写源文件，用户可以利用命令 \boldsymbol 自已定义一些经常使用的黑体数学符号。例如：

```
\newcommand{\bpi}{\boldsymbol{\pi}}
\newcommand{\binfty}{\boldsymbol{\infty}}
$$B_\infty+ \pi B_1 \sim
\mathbf{B}_{\binfty} \boldsymbol{+}
\bpi \mathbf{B}_{\boldsymbol{1}}$$
```

$$B_\infty+ \pi B_1 \sim \mathbf{B}_{\infty} \boldsymbol{+} \boldsymbol{\pi} \mathbf{B}_{\boldsymbol{1}}$$

表 5.20 AMSLaTeX 中数学模式字体命令

命 令	说 明
\mathbb	在数学公式中用来排印一种双轮廓的字母（只能用于大写字母），例如命令 $\mathbb {NQRZ}$，输出 $\mathbb{NQRZ}$（需要 amssymb 宏包）。
\mathfrak	排印 Euler Frakter 字母，如，$\mathfrak {E}=\mathfrak {mc}^2$，给出 $\mathfrak{E}=\mathfrak{mc}^2$（需要 amssymb 宏包）。
\boldsymbol	排版黑体数字和非字母的符号以及希腊字母（由宏包 amsbsy 定义）。
\pmb	用于排版 Poor man 型黑体数学符号。有些数学符号不存在黑体形式，此时可以用这条命令。例如命令 $\pmb {\oint }$ 和 \pmb {$\triangle $} 分别给出 $\pmb{\oint}$ 和 $\pmb{\triangle}$。
\text	在数学公式中用当前字体合适的字距排印正常的文字。例如命令 $E=mc^2\quad \text {(Einstein)}$ 给出 $E=mc^2 \quad \text{(Einstein)}$（由宏包 amsbsy 定义）。

在某些字体中，有些数学符号不存在黑体形式，因此对它们使用命令 \boldsymbol 就不起作用。此时可以利用命令 \pmb 来构造一种类似于通常黑体符号的 Poor man 型黑体符号，它实际上是将正常的非黑体符号重复打印几次，每次打印都有一个微小的位移。使用这种方式来获得黑体符号会自动调用 cmex 字库中的大算符和扩展符号，以及来自于字库 msam 和 msbm 的 $\mathcal{AMS}$ 额外数学符号。例如：

```
$$\frac{\partial w}{\partial u}
\pmb{\Bigg\vert}
\frac{\partial u}{\partial v}$$
```

$$\frac{\partial w}{\partial u}\pmb{\Bigg\vert}\frac{\partial u}{\partial v}$$

但是对于 cmex 中的大算符和扩展符号（如 $\sum, \prod$），由于不能很好的调整上限和下限的位置，用 \pmb 命令产生的黑体符号的效果并不是很好。因此，有必要使用 TeX 的 \mathop 命令。例如：

```
$$\sum_{j<P}\prod_\lambda \lambda R(r_j)
\qquad \mathop{\pmb{\sum}}_{x_j}
\mathop{\pmb{\prod}}_\lambda%
\lambda R(x_j)$$
```

$$\sum_{j<P}\prod_{\lambda}\lambda R(r_j) \qquad \sum_{x_j}\prod_{\lambda}\lambda R(x_j)$$

要想把某一数学公式中的所有符号（或尽可能多的符号，这依赖于所用的字体）都变为黑体，可以在这个公式中的最前面使用命令 \boldmath。

在标准的 LaTeX 中，命令 \mathbf{\hat{A}} 会在 $\mathbf{A}$ 上面产生一个黑体重音符号。但是要在 A 上面放一个重音符号就不能用命令 \mathcal{\hat{A}}，这是因为

\mathcal 字体没有自己的重音符号。在 amsmath 宏包中，改变字体的命令是这样定义的：当某种字体没有重音字符时（除了 \mathcal 字体外， \mathbb 和 \mathfrak 字体都没有重音），重音符号就从 \mathrm 字体中产生。

另外，在 amsmath 宏包中还提供了一些用来排版斜体大写希腊字母的命令，见表 5.21。

表 5.21 斜体大写希腊字母（需要 amsmath 宏包）

符号及其命令	符号及其命令	符号及其命令
$\varGamma$ \varGamma	$\varSigma$ \varSigma	$\varDelta$ \varDelta
$\varUpsilon$ \varUpsilon	$\varTheta$ \varTheta	$\varPhi$ \varPhi
$\varLambda$ \varLambda	$\varPsi$ \varPsi	$\varXi$ \varXi
$\varOmega$ \varOmega	$\varPi$ \varPi	

表 5.22 中列出了另外几种在数学公式中常用的字体及所需的宏包。

表 5.22 数学公式中几种常用字体

命 令	输出结果	所需宏包
\mathcal{ABCDEFG}	$\mathcal{ABCDEFG}$	amsmath
\mathscr{ABCDEFG}	$\mathscr{ABCDEFG}$	mathrsfs
\mathds{ABCDEFG1hk}	$\mathds{ABCDEFG1hk}$	dsfont
\mathocm{ABCDEFGabcd123}	$\mathocm{ABCDEFGabcd123}$	ocm
\mathbbm{ABCDEFGabcd12}	$\mathbbm{ABCDEFGabcd12}$	bbm

5.2.3 数学公式中的字体尺寸

在数学公式中 TeX 会根据公式周围文字的大小来自动选择合适的字体尺寸，比如上标和下标的尺寸就要比通常的字体小一些。但有时候我们仍需要按照自己的意愿来改变公式中某些字体的大小，此时可以使用下面 4 个字体尺寸格式命令：

```
\displaystyle, \textstyle, \scriptstyle, \scriptscriptstyle
```

\displaystyle 将公式按照独立显示公式中的字体大小来排版；\textstyle 将公式按照文本行公式中的字体大小来排版；\scriptstyle 按照通常上下标的大小来排版；而 \scriptscriptstyle 是按照更小一号的字体来排版。要注意改变数学公式的字体尺寸格式同时也会改变公式中求和限及积分限的排版方式。

5.3 数学公式的显示及对齐

amsmath 宏包提供了许多排版数学公式的环境，扩展了标准 LaTeX 系统排版数学公式的能力，使用户可以很方便地将数学公式按各种不同的方式对齐。这些环境包括：

equation	equation*	align	align*	gather
gather*	flalign	flalign*	multline	multline*
alignat	alignat*	split		

虽然标准 LaTeX 中排版数学公式的环境 eqnarray 仍可以使用，不过使用 align 环境或者使用 equation 环境并结合 split 环境来排版所得到的效果要好些。

除了 split 环境之外，上面每个环境都有不带星号和带星号两种形式。用不带星号的环境排版的数学公式会调用 LaTeX 的 equation 计数器自动生成公式编号，而带星号的环境则不产生编号。对于多行公式，如果只是部分行带编号而不希望另一些行含有编号，那么可以在不含编号的那些行的后面，换行符 \\ 之前，加上命令 \notag; 也可以利用命令 \tag{*label*} 产生的标签来覆盖原来的公式编号，其中 *label* 就是标签，它可以是任何文本，如 *, ii 等。在公式中用来覆盖公式编号的标签会被置于一对圆括号中。不过如果用命令 \tag* 来替代前面的 \tag，那么编号就是原来输入的文本且不带圆括号。命令 \tag 和 \tag* 也可以用在那些带星号的环境中。

通常情况下公式编号都是由 LaTeX 系统自动产生的，而且公式编号放置的位置也是由 LaTeX 系统决定的，系统对此已做了优化处理，因此应当尽量避免对编号位置进行直接操作。在 amsmath 中，当公式太长时，其编号有可能被下移或者上移至单独的一行。当然，如果这种情况频繁出现，那么就影响了版面的美观。amsmath 提供的命令 \raisetag 使我们能手工调整公式编号的垂直位置。比如命令 \raisetag{6pt} 将编号从通常的位置上移 6pt 的距离。不过这种调整应当属于精调范围，它应当在文稿的编辑基本结束，将要生成最后的 dvi 文件并打印输出文稿时来进行。

要注意 split 环境是一个特殊的环境，它只能用于除了 multline 和 multline* 之外的其他公式环境中来排版单个数学公式，而此公式又太长必须分成多行排版。因此，不能用

```
\begin{split} some formulas  \end{split}
```

这样的语句直接排版公式。排版具有对齐结构的多行公式时，LaTeX 源文件中必须用 & 这个字符来引导所要对齐的公式中的符号，但是与标准 LaTeX 的公式环境 eqnarray 不同，在对齐符号的后面不需要添加 &，否则就不能得到正确的排版结果。

5.3.1 单个公式

与标准 LaTeX 一样，环境 equation 用来排版单个带编号的数学公式。但 amsmath 宏包还提供了带星号的 equation* 环境，它同样用来排版单个数学公式，只是不带编号。下面是两个简单例子：

```
\begin{equation}
a+b=c
\end{equation}
```

$$a+b=c \tag{5.1}$$

```
\begin{equation*}
a+b=c
\end{equation*}
```

$$a+b=c$$

5.3.2 不对齐方式分裂公式

有些公式太长，需要将它分裂成多行来排版。multline 环境就是用来排版这种占多行的公式的。在默认的情况下此环境中的第 1 行居左排版，最后一行居右排版，中间的各行居中。在第 1 行的左边和最后一行的右边都会有一个由长度参数 \multlinegap 定义的缩格。参数命令 \multlinegap 的值可以用 LaTeX 的 \setlength 或者 \addtolength 命令来改变。如果在 \documentclass 的选项中使用了 fleqn，那么除了最后一行外所有行都靠左对齐，而最后一行则靠右排版，但向里有一个缩格。

与 equation 环境一样，multline 环境也只产生一个公式编号。因此，其他各行的后面不需要使用 \notag 或者 \nonumber 命令。默认的情况下，公式的编号是放在右边的（reqno 选项），并且编号在最后一行，但若在 \documentclass 的选项中使用了 leqno，则编号就被放在公式的左边并与第 1 行对齐。用 multline 环境来排版公式时，编号无法放在整个公式垂直位置的中间，而是与公式的最后一行对齐。例如：

```
\begin{multline}
A = a+b+c+d+e+f+ \\
    e+h+i+j+k+l+\\
    e+h+i+j+k+l+\\
    m+n+o+p+q+r+s+t
\end{multline}
```

$$\begin{aligned} A = {} & a+b+c+d+e+f+ \\ & e+h+i+j+k+l+ \\ & e+h+i+j+k+l+ \\ & m+n+o+p+q+r+s+t \end{aligned} \tag{5.2}$$

使用 multline 环境排版多行数学公式时，若想使中间的行居左或居右，则可以利

用命令 \shoveleft 和 \shoveright, 前者迫使中间的某一行居左排版，而后者迫使它居右排版。例如：

```
\begin{multline}
A = a+b+c+d+e+f \\
    \shoveleft{+e+h+i+j+k+l}+\\
    m+n+o+p+q+r+s+t
\end{multline}
```

$$\begin{aligned} A = a+b+c+d+e+f \\ +e+h+i+j+k+l+ \\ m+n+o+p+q+r+s+t \end{aligned} \tag{5.3}$$

5.3.3 对齐方式分裂公式

与 multline 环境一样，环境 split 也可以用来将一个一行排不下的长公式截成多行来排版。然而与 multline 环境不同的是，split 环境只能将此环境中的各行对齐，并且对齐符号需要用 & 来引导。不像 amsmath 宏包的其他公式结构，split 环境必须置于其他除了 multline 之外的公式环境中，因而它本身不产生公式编号。例如：

```
\begin{equation}\label{eq:barwq}
    \begin{split}
        H_c&=\frac1{2n}\sum^n_{l=0}(-1)^l(n-l)^{p-2}
        \sum_{l_1+\dots+l_n=l}\prod^p_{i=1}\binom{n_i}{l_i}\\
        &\quad\cdot[(n-1)-(n_i-l_i)]^{n_i-l_i}\cdot
        \Bigl[(n-l)^2-\sum^p_{j=1}(n_i-l_i)^2\Bigr]
    \end{split}
\end{equation}
```

$$\begin{aligned} H_c = \frac{1}{2n}\sum_{l=0}^{n}(-1)^l(n-l)^{p-2} \sum_{l_1+\cdots+l_n=l}\prod_{i=1}^{p}\binom{n_i}{l_i}\cdot \\ [(n-1)-(n_i-l_i)]^{n_i-l_i}\cdot\left[(n-l)^2-\sum_{j=1}^{p}(n_i-l_i)^2\right] \end{aligned} \tag{5.4}$$

5.3.4 不对齐的公式组

当有一组公式需要放在一起且不必考虑是否对齐时，可以使用 gather 环境。这个环境用于将一组连贯的公式以不对齐的方式排列起来，其中每一行都置于文本行的中间。gather 环境中的各个公式由换行符 \\ 分开。每一个公式都可以包含在 split 环境中。例如：

```
\begin{gather}
a_1=b_1+c_1 \\
\begin{split}
a_2=b_2+c_2+d_2+e_2+ \\ f_2+g_2+h_2
\end{split} \\
a_3=b_3+c_3+d_3+e_3
\end{gather}
```

$$a_1 = b_1 + c_1 \tag{5.5}$$

$$\begin{aligned} a_2 = b_2 + c_2 + d_2 + e_2 + \\ f_2 + g_2 + h_2 \end{aligned} \tag{5.6}$$

$$a_3 = b_3 + c_3 + d_3 + e_3 \tag{5.7}$$

5.3.5 对齐的公式组

如果要将一组公式对齐排版可以利用 align 环境。此时所要对齐的符号前面必须用 & 来引导，习惯上都是将二元关系符（如等号、不等号等）作为对齐符号。利用额外的 & 号，还可以把公式排成多列。例如：

```
\begin{align}
x&=y &  a&=b+c \\
x'&=y' & a'&=b'+c'\\
x''&=y'' & a''&=b''+c''
\end{align}
```

$$x = y \qquad a = b + c \tag{5.8}$$

$$x' = y' \qquad a' = b' + c' \tag{5.9}$$

$$x'' = y'' \qquad a'' = b'' + c'' \tag{5.10}$$

有时需要在公式中某一行的后面插入一些文字对该公式进行注解，此时可以利用命令 \text。该命令后面花括号中的文字被切换成左右模式来排版。比如下例中将注释的文字单独排成一列。

```
\begin{align*}
x&=y_1-y_3+y_5+\cdots
     && \text{by Theorem 1} \\
 &=y'\circ y^*
     && \text{by \eqref{eq:barwq}}\\
 &=z && \text{by definition}
\end{align*}
```

$$\begin{aligned} x &= y_1 - y_3 + y_5 + \cdots && \text{by Theorem 1} \\ &= y' \circ y^* && \text{by (5.4)} \\ &= z && \text{by definition} \end{aligned}$$

利用环境 alignat 还可以将公式分成若干列，并且可以明确地设置列与列之间的水平距离。这个环境带有一个参数，它的值是一个自然数，代表公式组包含的列数。例如下面的例子中列与列之间的间距设置为一个 \quad。

```
\begin{alignat*}{2}
x&=y_1-y_3+y_5+\cdots
      &\quad& \text{by Theorem 1} \\
 &=y'\circ y^*
      && \text{by \eqref{eq:barwq}}\\
 &=z && \text{by definition}
\end{alignat*}
```

$$\begin{aligned} x &= y_1 - y_3 + y_5 + \cdots && \text{by Theorem 1} \\ &= y' \circ y^* && \text{by (5.4)} \\ &= z && \text{by definition} \end{aligned}$$

5.3.6 公式中的块

与 equation 环境一样，排版多行公式的环境 gather, align 和 alignat 都创建了一个多行结构，其中每一行的宽度就是整个文本行的宽度，即 \linewidth。因此，不能用括号将这些行整个括起来。不过这几个环境都分别有自己的变体，即 gathered, aligned 和 alignedat，它们也都可以创建多行公式结构，其中每一行的宽度恰好就是其内容的宽度。因而可以利用这几个带 ed 的环境来建立公式中的块。当然公式中的块是可以加各种括号的。例如：

```
\begin{equation*}
\left.\begin{aligned}
      B' &= -\partial E,\\
      E' &= \partial B -4\pi j,
\end{aligned}\right\}
\qquad \text{Maxwell's equations}
\end{equation*}
```

$$\left.\begin{aligned} B' &= -\partial E, \\ E' &= \partial B - 4\pi j, \end{aligned}\right\} \qquad \text{Maxwell's equations}$$

因为只用了右边括号 \right\}，所以必须用带点的命令 \left. 与之相配。

类似于排版表格的 array 环境（见 99 页），gathered、aligned 和 alignedat 环境也都有一个选项。选项的值可以是 t 或者 b, 用来决定所建立的块的垂直位置。选项 t 表示顶部对齐，而选项 b 则表示底部对齐。利用这个方法，amsmath 宏包中还特别定义了一个 cases 环境，用来方便地排版数学公式中经常用到的 cases 结构。例如：

```
\begin{equation}
P_{j}=\begin{cases}
0, & \text{if $j$ is odd},\\
r!\,(-1)^{j/2}, &\text{if $j$ is even}.
\end{cases}
\end{equation}
```

$$P_j = \begin{cases} 0, & \text{if } j \text{ is odd}, \\ r!\,(-1)^{j/2}, & \text{if } j \text{ is even}. \end{cases} \tag{5.11}$$

5.3.7 公式中的换行和换页

在 amsmath 宏包提供的排版数学公式的环境中，与标准 LaTeX 一样也可以用命令 \\[*length*]（见 43 页）来获得额外的垂直空白。但是不允许在多行公式的行之间出现断页的情况，这是由于将一个公式分成两页排版可能使公式的意义变得不清楚。因此，应当由作者来决定是否允许在公式中间换页，或者在何处可以换页。如果作者有意在公式中间换页，则可以在换页处使用 amsmath 宏包提供的命令 \displaybreak，这条命令最好放在某一个换行符 \\ 之前。另外，这条命令与 \pagebreak 一样也有一个选项，选项的值可以是从 0 到 4 这几个数之一，表示允许换页的程度。\displaybreak[0] 意味着允许换页但尽量不要换页，而命令 \displaybreak[4] 则强迫换页。不给出选项的命令 \displaybreak 与 \displaybreak[4] 等价。

如果不想在个别的公式中使用命令 \displaybreak，而是允许在所有的多行公式中间换页，那么可以在源文件的导言部分使用命令 \allowdisplaybreaks[*n*]，选项 *n* 是 1 到 4 之间的整数，表示在所有公式中间允许换页的程度。当已经使用了命令 \allowdisplaybreaks 时，则与通常情况一样可以在多行公式的行之间使用 * 来阻止在此处换页。

注意，某些多行公式环境是将公式内容放置在一个不可分割的盒子之中，因此在这些公式中间，命令 \displaybreak 和 \allowdisplaybreaks 都不起作用。这些公式环境包括 split, aligned, gathered 和 \alignedat。

5.3.8 行与行之间的文字

命令 \intertext 用于在多行公式的行与行之间插入一两行较短的文本。这条命令的突出特点是此命令上下的公式仍保持原来的对齐方式。一般来说仅仅简单地将两组公式放在一起并不能保持这两组公式的对齐。命令 \intertext 只能紧跟着换行符 \\ 和 * 之后。请注意下面例子中“and”的位置：

```
\begin{align}
A_1 &= N_0(\lambda;\varOmega')
      -\phi(\lambda;\varOmega'),\\
A_2 &= \phi(\lambda;\varOmega')
      -\phi(\lambda;\varOmega),\\
\intertext{and}
A_3 &= \mathcal{N}(\lambda;\varOmega).
\end{align}
```

$$A_1 = N_0(\lambda;\Omega') - \phi(\lambda;\Omega'), \tag{5.12}$$

$$A_2 = \phi(\lambda;\Omega') - \phi(\lambda;\Omega), \tag{5.13}$$

and

$$A_3 = \mathcal{N}(\lambda;\Omega). \tag{5.14}$$

这里的“and”与公式之外的文字左边对齐。

5.3.9 公式的编号

在 LaTeX 中如果要使数学公式按节（section）进行编号，比如说第 1 节的节序号是 1, 其中的公式编号分别为 (1.1), (1.2) 等，第 2 节的节序号是 2, 其中的公式编号分别为 (2.1), (2.2) 等，那么可以在源文件的导言部分按下面的方法

```
\renewcommand{\theequation}{\thesection.\arabic{equation}}
```

来重新定义命令 \theequation，然后在源文件的正文部分中每一条命令 \section 之后使用如下命令

```
\setcounter{equation}{0}
```

将计数器 equation 的值重新设置为零。这种方法的确是有效可行的，只是稍嫌麻烦。为此，amsmath 宏包提供了命令 \numberwithin，我们只需在源文件的导言部分输入命令

```
\numberwithin{equation}{section}
```

就可以将公式编号的前面附加节号，并且在每一节的开始公式编号会自动设置为零。实际上，命令 \numberwithin 不仅可以用于计数器 equation, 也可以应用于任何其他的计数器。

可以使用命令 \eqref 对公式编号进行交叉引用。比如，在某个编号的数学公式中已经用命令 \label{abc} 设置了标签并且此公式的编号为 (1.2)，则命令 \ref{abc} 产生 1.2, 而命令 \eqref{abc} 产生 (1.2)。

amsmath 宏包还提供了一个 subequations 环境来排版一组具有某种公共性质的公式。这个环境中的所有公式都被赋予子公式编号，也就是说当这个环境之前的公式编号是 (4.8) 时，那么此环境中的公式按照 (4.9a), (4.9b), (4.9c), $\cdots$ 的顺序编号。紧跟在命令 \begin{subequations} 后面的 \label 命令会产生一个引用主公式号 (4.9) 的标签。例如：

```
\begin{subequations}\label{subequation}
\begin{align}
x+y&=1 \label{sub1}\\
y+z&=2 \\
z+x&=3
\end{align}
\end{subequations}
```

$$x+y=1 \tag{5.15a}$$

$$y+z=2 \tag{5.15b}$$

$$z+x=3 \tag{5.15c}$$

在正文中输入 \eqref{subequation} 时可得到 (5.15)，当输入 \eqref{sub1} 时可得到

(5.15a)。记录主公式编号（在上面的例子中是 (5.15)）的计数器是 parentequation，而记录子公式的编号（在上面的例子中是 a、b、c）的计数器是 equation。默认情况下，子公式的编号是小写英文字母，但也可重新定义为其他的形式。例如下面的语句：

```
\begin{subequations}
\renewcommand{\theequation}{\theparentequation\roman}
 ......
\end{subequations}
```

可以将此环境中的子公式编号重新定义为小写罗马字母。

5.3.10 精调数学公式中的间距

虽然 TeX 在一般情况下都能很好地处理数学公式中的间距，但有时也还是有必要对其中的一两个间距进行细微地调整。表 5.23 中提供了数学公式中常用的一些调整间距的命令。其中的命令以及它们的缩略形式都是所谓 Robust 命令，可以用于数学模式之外。

表 5.23 数学模式中的间距命令

正 间 距			负 间 距		
缩略命令	例子	命 令	缩略命令	例子	命 令
\,	$x\,x$	\thinspace	\!	$x\!x$	\negthinspace
\:	$x\:x$	\medspace		$x\!\!x$	\negmedspace
\;	$x\;x$	\thickspace		$x\!\!\!x$	\negthickspace
	$x\quad x$	\quad			
	$x\qquad x$	\qquad			

例如，下面两个积分表达式中 $f(x)$ 与 $\mathrm{d}x$ 的间距有微小区别：

```
$$\int^b_af(x)\mathrm{d}x\quad
\int^b_af(x)\,\mathrm{d}x$$
```

$$\int_a^b f(x)\mathrm{d}x \quad \int_a^b f(x)\,\mathrm{d}x$$

用户还可以使用命令 \mspace 来进一步调整数学公式中的间距。这条命令的参数是 LaTeX 提供的数学长度单位（math unit）。一个数学长度单位，即一个 mu 等于 1/18 em。于是，可以用命令 \quad=\mspace{-18mu} 来得到一个 \quad 的负间距。

5.4 各种数学公式和结构

5.4.1 矩阵

标准 LaTeX 中的 array 环境可以用来排版数学公式中的一般矩阵结构。amsmath 宏包在 array 环境的基础上又提供了一些新的环境用来方便地排版各种常用的矩阵。这些环境是：pmatrix，用于排版圆括号的矩阵；bmatrix，用于排版方括号的矩阵；Bmatrix，用于排版带花括号的矩阵；vmatrix，用于排版两边各用一条垂直线括起来的矩阵（如行列式）；Vmatrix，用于排版两边各用两条垂直线括起来的矩阵。另一个不带定界符的矩阵环境是 matrix。例如：

```
\begin{gather*}
\begin{matrix}a& b\\c& d\end{matrix}%
\quad
\begin{pmatrix}a& b\\c& d\end{pmatrix}%
\quad
\begin{bmatrix}a& b\\c& d\end{bmatrix}\\
\begin{vmatrix}a& b\\c& d\end{vmatrix}%
\quad
\begin{Vmatrix}a& b\\c& d\end{Vmatrix}
\end{gather*}
```

$$\begin{matrix}a & b\\ c & d\end{matrix} \quad \begin{pmatrix}a & b\\ c & d\end{pmatrix} \quad \begin{bmatrix}a & b\\ c & d\end{bmatrix}$$

$$\begin{vmatrix}a & b\\ c & d\end{vmatrix} \quad \begin{Vmatrix}a & b\\ c & d\end{Vmatrix}$$

amsmath 宏包所提供的这些 matrix 类环境有一个与 array 环境不同之处，就是它们不需要在环境中特别指明矩阵的行列数。不过在默认的情况下 matrix 类的矩阵环境要求矩阵的行数和列数不超过 10。这个最大值是由计数器 MaxMatrixCols 决定的，因此，也可以用 LaTeX 的 \setcounter 或 \addtocounter 来重新定义这个最大值。当然这个最大值定义的越大，消耗的内存也越多。

amsmath 宏包还提供了一个名为 smallmatrix 的环境用来在文本行中排版一些类似于 $\left(\begin{smallmatrix}a & b\\ c & d\end{smallmatrix}\right)$ 这样的小矩阵。注意，这个环境本身是不带定界符的，因此必要时必须另外添上。例如上面一行的小矩阵可以用命令

```
\bigl(\begin{smallmatrix}
a & b \\
c & d
\end{smallmatrix}\bigr)
```

来获得。这里分别用 \bigl(和 \bigr) 加上了左圆括号和右圆括号。

另外，在 matrix 类的矩阵环境中还可以利用下面的命令

\hdotsfor[m]{n}

来产生跨越 n 行的连续黑点，其中参数 n 是自然数，表示黑点所要跨越的行数，选项 m 是一个因子，表示其中相邻两个黑点之间的距离是通常两个黑点之间的距离的多少倍，其默认值是 1，即与通常两个黑点之间的距离相等。例如：

```
\begin{equation*}
\begin{matrix}
a & b & c & d \\
e & \hdotsfor{3}\\
\hdotsfor{4}
\end{matrix}
\end{equation*}
```

$$\begin{matrix} a & b & c & d \\ e & \cdots & & \\ \cdots & & & \end{matrix}$$

下面是另一种省略了矩阵中部分元素的例子：

```
\begin{equation}
\begin{pmatrix}
a_{11} & a_{12} & \dots & a_{1n} \\
a_{21} & a_{22} & \dots & a_{2n} \\
\vdots & \vdots &       & \vdots \\
a_{n1} & a_{n2} & \dots & a_{nn}
\end{pmatrix}
\end{equation}
```

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \tag{5.16}$$

5.4.2 省略点

在数学公式中经常使用命令 \ldots 和 \cdots 来排印省略意义的连续三个点。\ldots 排印的点与逗号对齐，而 \cdots 排印的点居于垂直中间位置。amsmath 宏包还提供了下面几个排印省略点的命令来适应不同的情况：

\vdots　用来排印垂直的省略点。

\ddots　用来排印向左倾斜的省略点。

\dotsc　用来排印跟在逗号后面的省略点。

\dotsb　用来排印跟在二元算符或关系符后面的省略点。

\dotsm　用于乘积中的省略点。

\dotsi　用来排印跟在积分号后面的省略点。

\dotso　用于其他情况下的省略点。

例如：

```
Then we have the series $A_1, A_2,
\cdots$, the regional sum $A_1
+A_2 +\dotsb $, the orthogonal
product $A_1 A_2 \dotsm $, and
the infinite integral
\[\int_{A_1}\int_{A_2}\dotsi.\]
```

Then we have the series $A_1, A_2, \cdots$, the regional sum $A_1 + A_2 + \cdots$, the orthogonal product $A_1 A_2 \cdots$, and the infinite integral

$$\int_{A_1}\int_{A_2}\cdots.$$

5.4.3 数学符号的重音

在标准 LaTeX 中，在已经有一个重音的数学符号上面再加一个重音，输出效果通常不是很好。amsmath 宏包提供了改进的方法来排印有两个重音的数学符号。例如：

```
\begin{gather*}
\Acute{\Acute{A}}\qquad \Bar{\Bar{B}}
\qquad\Breve{\Breve{C}} \qquad
\Check{\Check{D}}\\
\Ddot{\Ddot{E}}\qquad \Dot{\Dot{F}}
\qquad\Grave{\Grave{G}}\qquad
\Hat{\Hat{H}} \\
\Tilde{\Tilde{I}}\qquad \Vec{\Vec{J}}
\end{gather*}
```

$$\acute{\acute{A}} \qquad \bar{\bar{B}} \qquad \breve{\breve{C}} \qquad \check{\check{D}}$$

$$\ddot{\ddot{E}} \qquad \dot{\dot{F}} \qquad \grave{\grave{G}} \qquad \hat{\hat{H}}$$

$$\tilde{\tilde{I}} \qquad \vec{\vec{J}}$$

这些有两个重音的符号会减慢 LaTeX 的编译速度。当源文件中有许多二重重音符号时，可以加载 amsxtra 宏包并且在导言部分使用命令 \accentedsymbol 来定义出现的二重重音。这样，LaTeX 就会将所有二重重音先装在一个盒子中，从而加快编译速度。\accentedsymbol 的使用方法与 \newcommand 类似。例如：

```
\accentedsymbol{\Ahathat}%
{\Hat{\Hat A}}
\accentedsymbol{\dbardot}%
{\Dot{\Bar\delta}}
This is a double hat $\Ahathat$
and this $\dbardot$ a delta with
a bar and a dot.
```

This is a double hat $\hat{\hat{A}}$ and this $\dot{\bar{\delta}}$ a delta with a bar and a dot.

有些重音符号具有加宽的形式，如命令 $\widehat{xy}$ 和 $\widetilde{xy}$ 分别输出 $\widehat{xy}$ 和 $\widetilde{xy}$. 由于这些加宽的重音其长度不能超过一个最大值，所以对于特别长的重音符号最好还是用其他形式来表示。比如可以用 $(ABCD)\hat{}$ 来代替 $\widehat{ABCD}$。amsmath 宏包中有一些命令使我们能方便地做到这些。

```
\begin{gather*}
(ABCD)\sphat \qquad (ABCD)\spcheck\\
(ABCD)\sptilde \qquad (ABCD)\spdot\\
(ABCD)\spddot \qquad (ABCD)\spdddot\\
(ABCD)\spbreve
\end{gather*}
```

$$(ABCD)\sphat \qquad (ABCD)\spcheck$$
$$(ABCD)\sptilde \qquad (ABCD)\spdot$$
$$(ABCD)\spddot \qquad (ABCD)\spdddot$$
$$(ABCD)\spbreve$$

5.4.4 不分断的连字符

数学中常常出现一些不能分断的连字符，如 p-adic, n-dimensional, page 1–9 等。如果不加以保护，有时 TeX 系统就会将连字符与它后面的词分开。为此，amsmath 宏包提供了命令 \nobreakdash。紧跟在这条命令后面的连字符就不会与连字符后面的词分开了。于是前面的三个例子就可以分别用命令 p\nobreakdash-adic, n\nobreakdash-dimensional 和 page 1\nobreakdash--9 来得到。当这种情形频繁出现时，建议重新定义一些简单的命令。如果定义下面三条命令

```
\newcommand{\p}{$p$\nobreakdash}
\newcommand{\Ndash}{\nobreakdash--}
\newcommand{\n}[1]{$n$\nobreakdash-\hspace{0pt}}
```

那么前面三个例子就可以分别用 \p-adic, page 1\Ndash9 和 \n dimensional 来得到。其中最后一个定义展示了如何使连字符与紧跟在它之后的词不被分开，同时使后面的连字符具有通常的意义。

5.4.5 根号

通常情况下可以使用命令 \sqrt[\beta]{k} 来得到根号：$\sqrt[\beta]{k}$。但这样得到的方根 β 的位置看起来不是很合适。为此 amsmath 宏包提供了命令 \leftroot{n} 和 \uproot{n} 来调整根号中方根的位置，其意义为分别将方根向左和向上移动 n 个单位（n 为负数则表示向反方向移动）。例如：

```
$\sqrt[\leftroot{-2}\uproot{2}\beta]{k}$
```

$$\sqrt[\leftroot{-2}\uproot{2}\beta]{k}$$

5.4.6 带方框的数学公式

可以利用命令 \boxed 将数学公式置于方框中[1]，此命令的作用类似于一般 LaTeX

[1] fancybox 宏包提供了几个环境和命令可以将公式连同编号一起放在方框中。

中的命令 \fbox，但它只是在数学模式中使用。例如：

```
\begin{equation}
\boxed{\eta\leq C(\delta(\eta)+
\varLambda_M(0,\delta))}
\end{equation}
```

$$\boxed{\eta \leq C(\delta(\eta) + \varLambda_M(0,\delta))} \tag{5.17}$$

5.4.7 上置箭头和下置箭头

标准 LaTeX 已有命令 \overleftarrow 和 \overrightarrow 用来在符号的上方排印左向箭头和右向箭头。amsmath 宏包提供了额外的命令使得在符号的下方也能排印箭头。例如：

```
\begin{align*}
\overleftarrow{ABC}&=
\underleftarrow{ABC}\\
\overrightarrow{ABC}&=
\underrightarrow{ABC}\\
\overleftrightarrow{ABC}&=
\underleftrightarrow{ABC}
\end{align*}
```

$$\begin{aligned}
\overleftarrow{ABC} &= \underleftarrow{ABC}\\
\overrightarrow{ABC} &= \underrightarrow{ABC}\\
\overleftrightarrow{ABC} &= \underleftrightarrow{ABC}
\end{aligned}$$

这些箭头命令即使用在上标或下标中，输出的箭头长度也能很好地自动调整，如符号 $\int_{\overrightarrow{ab}} f(t)\,\mathrm{d}t$ 可以用 $\int_{\overrightarrow{ab}}f(t)\,\mathrm{d}t$ 得到。

5.4.8 扩展箭头

数学中经常需要在箭头的上方或者下方显示某些符号和文字，而这些符号和文字的长度有长有短。此时可使用命令 \xleftarrow 和 \xrightarrow 来产生箭头。这两条命令都带有参数和选项分别表示箭头的上方符号和下方符号，箭头的长度会自动适合其上下方的符号。例如：

```
$$ 0 \xleftarrow[\zeta]{\alpha} F
\xrightarrow[T]{n+\mu-1} E $$
```

$$0 \xleftarrow[\zeta]{\alpha} F \xrightarrow[T]{n+\mu-1} E$$

5.4.9 上置符号、下置符号和旁置符号

在标准 LaTeX 中，命令 \stackrel 可以用来将一个符号放置在另一个二元算符的上面。amsmath 宏包提供了功能更强大的命令 \overset 和 \underset，用它们可以将

一个符号放置在另一个符号上方和下方，而不必管它是否是二元算符。例如：

```
$$\overset{*}{X} \qquad
 \underset{*}{X} \qquad
 \overset{a}{\underset{b}{X}}$$
```

$$\overset{*}{X} \qquad \underset{*}{X} \qquad \overset{a}{\underset{b}{X}}$$

另一条命令 \sideset 具有相当特别的作用，用它可以将符号放置在一些大操作符如 $\sum$, $\prod$ 等符号的上、下标位置。如果仅仅在 $\sum$ 的上标位置放置一个符号如“'”，则只需按照下面方法利用命令 \nolimits 就可以了。

```
$$\sum\nolimits'E_n.$$
```

$$\sum\nolimits' E_n.$$

但如果要在一个有求和限的求和号的上标位置放置一个符号就不那么简单了。此时可以使用命令 \sideset, 例如：

```
$$ \sideset{_1^2}{_3^4}\prod\qquad
\sideset{}{'}\sum_{0\le i\le m}
E_i\beta x $$
```

$$\sideset{_1^2}{_3^4}\prod \qquad \sideset{}{'}\sum_{0\le i\le m} E_i\beta x$$

5.4.10 smash 命令

在 TeX 中 \smash 命令返回盒子的内容，并将盒子的高度和深度改为零。这条命令常被用来调整公式中某一部分相对于周围部分的位置。amsmath 宏包为此命令增添了两个选项 t 和 b。选项 t 使命令 \smash 只将盒子的高度改为零而保持其自然深度，选项 b 只改变盒子的深度而保持其自然高度。请比较下面两个例子：

```
$$X_j=(1/\sqrt{\smash[b]{\lambda_j}})X_j'
\qquad X_j=(1/\sqrt{\lambda_j})X_j'$$
```

$$X_j=(1/\sqrt{\smash[b]{\lambda_j}})X_j' \qquad X_j=(1/\sqrt{\lambda_j})X_j'$$

上面第一个例子展示了如何用命令 \smash 来限制根号的深度。

5.4.11 text 命令

命令 \text 的主要用处是在公式中排印文字或短语。虽然这个功能也可以用命令 \mbox 来实现，但命令 \text 用起来方便一些，而且在下标中使用这条命令时，它会自动选择合适的字体尺寸。例如：

```
$$F_{\text{word}}=0 \text{ and }
G_{\text{word}}=1$$
```

$$F_{\text{word}}=0 \text{ and } G_{\text{word}}=1$$

5.4.12 函数名

为了与数学公式中的字母符号（通常是斜体）区分开来，公式中的常用函数名习惯上采用直立的罗马字型。LaTeX 中已经定义了一些常用的函数名，如 123 页中表 5.6 所示。amsmath 宏包还增加了下面几个函数名：

\injlim	$\injlim$	\varinjlim	$\varinjlim$	\varlimsup	$\varlimsup$
\projlim	$\projlim$	\varprojlim	$\varprojlim$	\varliminf	$\varliminf$

因为数学文献中会不断出现新的函数名，所以 amsmath 宏包提供了一般的定义函数名的方法。比如在导言部分可以用

```
\DeclareMathOperator{\xxx}{xxx}
```

来定义一个类似于 \sin 的函数名 \xxx。此时数学公式中的命令 \xxx 会用正确的字体产生 xxx，而且其前后会自动添加必要的空白。比如，A\xxx B 输出 $A\operatorname{xxx}B$，而不是 $AxxxB$。如果要定义像 lim、sup、max 这些可带有上标或下标的函数名，则可以用带星号的 \DeclareMathOperator* 定义。例如，在导言部分输入下面的内容

```
\DeclareMathOperator*{\doublesum}{\sum\sum}
```

就可以在正文部分排版下面的例子：

```
$$\doublesum_{i,j=1}$$
```

$$\doublesum_{i,j=1}$$

另有一条在公式中使用的命令 \operatorname，使得

```
\operatorname{abc}
```

的作用等同于用 \DeclareMathOperator 定义的 \abc。这条命令有时在构造一些复杂的数学记号时是有用的。同样，用 \operatorname* 可以得到带上标或下标的函数名。

5.4.13 模及相关符号

命令 \mod, \bmod, \pmod 及 \pod 用来排印模符号及相关的一些符号。在标准 LaTeX 中也有 \bmod 和 \pmod 命令，但使用 amsmath 宏包时非单独显示的公式中的 \pmod 命令涉及的空白会被调整到更小。某些作者喜欢用 \mod 和 \pod 来代替命令

\pmod。命令\mod 省略了圆括号而命令 \pod 则省略了“mod”但保留圆括号。例如：

```
\begin{equation}
\gcd(k,l \bmod k)
\end{equation}
\begin{align}
u & \equiv v+1 \pmod{n^2} \\
u & \equiv v+1 \mod{n^2} \\
u & \equiv v+1 \pod{n^2}
\end{align}
```

$$\gcd(k, l \bmod k) \tag{5.18}$$

$$\begin{aligned} u &\equiv v+1 \pmod{n^2} && (5.19)\\ u &\equiv v+1 \mod{n^2} && (5.20)\\ u &\equiv v+1 \pod{n^2} && (5.21) \end{aligned}$$

5.4.14 多行上标或下标

利用命令 \substack 可以构造具有多行的上标或下标。这条命令所构造的多行上标或下标是中心对齐的。例如：

```
$$ \sum_{\substack{0\le i\le m\\
                   0<j<n}}P(i,j)$$
```

$$\sum_{\substack{0\le i\le m\\ 0<j<n}} P(i,j)$$

更一般地，可以使用环境 subarray 来构造多行上标或下标，且可以让多行上标或下标按不同的方式对齐。例如：

```
$$\sum_{\begin{subarray}{l}
i\in\Lambda \\ 0<j<n
\end{subarray}}P(i,j)$$
```

$$\sum_{\begin{subarray}{l} i\in\Lambda \\ 0<j<n \end{subarray}} P(i,j)$$

其中的选项 l 代表左对齐，它可以被换为 c 和 r，分别表示中心对齐和右对齐。

5.4.15 求和限和积分限的位置

默认情况下，求和类符号（如 $\sum, \prod$ 等）的上限和下限分别被置于“displaylimits”位置，即在单独显示的公式中求和类符号的上限和下限分别被置于符号的上方和下方，但是在文本行中，为了使一行的上下不出现过多的空白，上限和下限就被置于符号的旁边。对于积分类符号而言，其上限和下限总是放在旁边。不过可以在控制符（即^ 和 _）的前面加上命令 \limits 或 \nolimits 来改变上下限的位置。例如：

```
$$\sum^n_{j=1}a_j\
\sum\nolimits^n_{j=1}a_j\
\int^b_af(x)\,\mathrm{d}x\
\int\limits^b_af(x)\,\mathrm{d}x$$
```

$$\sum_{j=1}^{n} a_j \quad \sum\nolimits_{j=1}^{n} a_j \quad \int_a^b f(x)\,\mathrm{d}x \quad \int\limits_a^b f(x)\,\mathrm{d}x$$

5.4.16 多重积分号

多重积分号可以用 \iint, \iiint, \iiiint 等命令来得到。无论是在文本行中还是在单独显示的公式中，积分号之间的距离都会自动调整的很好。命令 \idotsint 是多重积分号的扩展，它输出两个积分号并且中间有 3 个点。例如：

```
\begin{align*}
&\iint\limits_Vf(x,y)
\,\mathrm{d}x\,\mathrm{d}y\qquad
\iiint\limits_Vf(x,y,z)\,\mathrm{d}x
\,\mathrm{d}y\,\mathrm{d}z\\
&\idotsint\limits_Vf(x_1,\cdots,x_n)
\,\mathrm{d}x
\end{align*}
```

$$\iint\limits_V f(x,y)\,\mathrm{d}x\,\mathrm{d}y \qquad \iiint\limits_V f(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$$

$$\int\limits_V \cdots \int f(x_1,\cdots,x_n)\,\mathrm{d}x$$

5.4.17 分数及相关结构

在标准 LaTeX 中分数可以由命令 \frac{}{} 得到。这条命令有两个参数，它们分别被置于两对花括号中，第一对括号中的参数表示分子，第二对括号中的参数表示分母。为了方便排版，amsmath 宏包还提供了命令 \dfrac 和 \tfrac 分别代表显示形式的分数（\displaystyle\frac ⋯ ）和文本形式的分数（\textstyle\frac ⋯ ）。文本形式的分数在尺寸上与文本行中的分数相同，而显示形式的分数在尺寸上则与独立显示公式中的分数相同。例如：

```
\begin{equation}
\frac{1}{k}\ln2\quad\tfrac{1}{k}\ln2
\end{equation}and $\sqrt{\frac{1}{k}\ln2}
\quad\sqrt{\dfrac{1}{k}\ln2}$
```

$$\frac{1}{k}\ln 2 \quad \tfrac{1}{k}\ln 2 \tag{5.22}$$

and $\sqrt{\frac{1}{k}\ln 2} \quad \sqrt{\dfrac{1}{k}\ln 2}$

与分数结构相关，amsmath 宏包提供了命令 \binom 来排印形如 $\binom{n}{k}$ 的组合数。此命令也有两个变体 \dbinom 和 \tbinom，分别用来排版显示形式的组合数和文本形式的组合数。例如：

```
\begin{equation}
\binom{k}{1}2^k+\tbinom{k}{2}3^k
\end{equation}and $\binom{k}{1}2^k+
\dbinom{k}{2}3^k$
```

$$\binom{k}{1}2^k+\tbinom{k}{2}3^k \tag{5.23}$$

and $\binom{k}{1}2^k+\dbinom{k}{2}3^k$

命令 \frac, \binom 及它们的各种变体都可以由一个更一般的命令 \genfrac 来定义，它的使用语法是：

```
\genfrac{左定界符}{右定界符}{长度}{格式}{分子}{分母}
```

这条命令有 6 个参数，其中最后两个参数分别是分子和分母，前面两个分别是左右定界符，第 3 个参数是一个长度，代表分数线的粗细，第 4 个是 0 至 3 这 4 个数之一，分别表示 \displaystyle, \textstyle, \scriptstyle 和 \scriptscriptstyle 这 4 种显示格式。如果第 3 个参数是空的，则分数线具有通常粗细。命令 \frac, \tfrac 和 \binom 就是按照下面的方法来定义的：

```
\newcommand{\frac}[2]{\genfrac{}{}{}{}{#1}{#2}}
\newcommand{\tfrac}[2]{\genfrac{}{}{}{1}{#1}{#2}}
\newcommand{\binom}[2]{\genfrac{(}{)}{0pt}{}{#1}{#2}}
```

5.4.18 连分数

amsmath 宏包提供的命令 \cfrac 可用来排版连分数，其输出效果要比标准 LaTeX 中的命令 \frac 的输出效果好一些。例如：

```
\begin{equation}
\cfrac{1}{\sqrt{2}+
 \cfrac{1}{\sqrt{3}+
  \cfrac{1}{\sqrt{4}+
   \cfrac[l]{1}{\sqrt{5}+
    \cfrac[r]{1}{\sqrt{6}+\dotsb
}}}}}
\end{equation}
```

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{3}+\cfrac{1}{\sqrt{4}+\cfrac{1}{\sqrt{5}+\cfrac{1}{\sqrt{6}+\cdots}}}}} \tag{5.24}$$

命令 \cfrac 还有一个可选项，其值可以取字母 l 或 r，分别表示分子与分数线左对齐和右对齐，上面的例子已展示了这一点。

5.4.19 交换图表

对于特别复杂的交换图表可以利用功能较强大的宏包（如 kuvio 宏包和 XY-pic 宏包）来排版，也可以利用标准 LaTeX 中的 picture 环境来排版。而对于较简单的不含有对角箭头的交换图表利用 AMSLaTeX 宏包套件中的 amscd 宏包来排版还是很方便的。此宏包提供了 CD 环境用来排版交换图表。例如：

```
\newcommand{\End}{\operatorname{End}}
$$\begin{CD}
S^W\otimes T @>j>> T\\
@VVV          @VV{\End P}V\\
(S\otimes T)/I @= (Z\otimes T)/J
\end{CD}$$
```

$$\begin{array}{ccc} S^W \otimes T & \xrightarrow{j} & T \\ \downarrow & & \downarrow{\scriptstyle \operatorname{End} P} \\ (S \otimes T)/I & = & (Z \otimes T)/J \end{array}$$

在 CD 环境中，命令 @>>>, @<<<, @VVV 和 @AAA 分别输出向右、向左、向下和向上的箭头。对于横向的箭头，介于第 1 个 > 或 < 和第 2 个 > 或 < 之间的内容被置于箭头上方，介于第 2 个 > 或 < 和第 3 个 > 或 < 之间的内容则被置于箭头的下方。类似地，对于垂直箭头，介于第 1 个 V 或 A 和第 2 个 V 或 A 之间的内容被置于箭头左边，介于第 2 个 V 或 A 和第 3 个 V 或 A 之间的内容则被置于箭头的右边。另外，命令 @= 和 @| 分别输出横向的和垂直的双线。

5.4.20 定界符尺寸

要使定界符的尺寸随着其中内容的大小而改变，可以在左定界符的前面加上命令 \left，在右定界符的前面加上命令 \right。但这样做有时也不是很合适。首先是对于包含有上下限的求和类符号的定界符，其高度可能显得太高。其次对于两对嵌套的定界符，\left 和 \right 可能并不改变定界符的尺寸。第三，在文本行内的公式中使用 \left 和 \right 有可能使公式上下出现过多的空白。例如：

```
$$\left[\sum_ia_i\right]^2 \quad
\left(a-\left(b+c\right)\right)$$
```

$$\left[\sum_i a_i\right]^2 \quad (a-(b+c))$$

为了更好地控制公式中的定界符，可以利用 TeX 中的命令 \big, \Big, \bigg 和 \Bigg。这些命令用于定界符的前面，将定界符尺寸逐渐增大。另外，这几条命令还有变体，如 \bigl 和 \bigr 分别用于左定界符和右定界符的前面。在标准 LaTeX 中，这几条命令产生的定界符的尺寸是固定的，并不随周围字体大小的改变而变化。如果使用 amsmath 宏包，那么定界符的尺寸就随周围字体大小的改变而改变。例如：

```
$$\Bigl[\sum_ia_i\Bigr]^2 \quad
\bigl(a-(b+c)\bigr)$$ \large
$$\Bigl[\sum_ia_i\Bigr]^2 \quad
\bigl(a-(b+c)\bigr)$$
```

$$\Big[\sum_i a_i\Big]^2 \quad \big(a-(b+c)\big)$$

$$\Bigg[\sum_i a_i\Bigg]^2 \quad \Big(a-(b+c)\Big)$$

5.5 定理环境的扩展

在 3.14.3 节中我们已经介绍了利用标准 LaTeX 的 \newtheorem 命令来定义一个定理型表述环境。但是在标准 LaTeX 中这样定义的定理表述环境的格式是固定的。为了适应不同数学杂志对定理格式的不同要求，我们在本节中详细介绍如何利用 AMSLaTeX 中的 amsthm 宏包来定义新的定理格式环境。

5.5.1 定理格式

在 amsthm 宏包中，命令 \newtheorem 的使用方法与标准 LaTeX 完全相同，在这里可以用命令

\theoremstyle{*style*}

来选择定理格式。比如，下面的命令

```
\theoremstyle{plain}    \newtheorem{Cor}{Corollary}
\theoremstyle{remark}   \newtheorem{Rem}{Remark}
```

定义了 plain 格式的定理环境 Cor 用来排版 Corollary，和 remark 格式的定理环境 Rem 用来排版 Remark。目前 amsthm 宏包中已经定义了下面三种定理格式：

palin　　模拟标准 LaTeX 中的定理格式，定理名用黑体（\bfseries），定理内容用意大利斜体（\itshape）。

definition　定理名用黑体，定理内容用罗马字体。

remark　　定理名用意大利斜体，定理内容用罗马字体。

有一种不常用的定理格式，它的定理序号放在定理名的前面而不是放在定理名后面。要得到这种格式的定理环境，可以在 \newtheorem 命令的前面放置命令 \swapnumbers。例如：

```
\theoremstyle{remark} \swapnumbers \newtheorem{Rem}{Remark}
```

5.5.2 定理格式举例

假设在导言部分包含下面的命令：

```
\theoremstyle{definition} \newtheorem{Defi}{Definition}
\theoremstyle{plain}  \newtheorem{theorem}{Theorem}
\theoremstyle{remark}  \newtheorem{Rem}{Remark}
\theoremstyle{plain} \swapnumbers  \newtheorem{Note}{Note}
```

则可以得到下面例子中的输出效果：

```
\begin{Defi}
This is a sentence typeset in the
theorem environment \texttt{Defi}.
\end{Defi}
```

Definition 1. This is a sentence typeset in the theorem environment Defi.

```
\begin{theorem}
This is a sentence typeset in the
theorem environment \texttt{theorem}.
\end{theorem}
```

Theorem 2. *This is a sentence typeset in the theorem environment theorem.*

```
\begin{Rem}
This is a sentence typeset in the
theorem environment \texttt{Rem}.
\end{Rem}
```

Remark 1. This is a sentence typeset in the theorem environment Rem.

```
\begin{theorem}[Ben User]
This is a sentence typeset in the
theorem environment \texttt{Thm}.
\end{theorem}
```

Theorem 3 (Ben User). *This is a sentence typeset in the theorem environment Thm.*

```
\begin{Note}
This is a sentence typeset in the
theorem environment \texttt{Note}.
\end{Note}
```

1 Note. *This is a sentence typeset in the theorem environment Note.*

5.5.3 新的定理格式

对于排版一般科技文稿来说，现有的几种定理格式已经基本够用了，但也有少数杂志使用一些不同的定理格式，特别是许多中文杂志所使用的定理格式与西文杂志不同。

因此，也有必要了解如何定义新的定理格式。在 LaTeX2e Tools 宏包套件中的 theorem 宏包以及它的一个扩展宏包 ntheorem 之中，都定义了一些高端命令用于改变定理格式。在 amsthm 宏包中提供了下面更为一般的命令来定义新的定理格式。

```
\newtheoremstyle{名称}{上空白 }{下空白}{主字体}%
                      {缩格}{头字体}{符号}{间距}{补充}
```

其中

名称　是所定义的定理格式的名称。

上空白　是定理环境与上方文字的额外空白。

下空白　是定理环境与下方文字的额外空白。

主字体　设置定理内容的字体。

缩格　定理名的缩进空白距离。省略时表示不缩进，而 \parindent 则表示与通常段落的缩进一样。

头字体　设置定理名所用的字体。

符号　定理头（定理名加序号）后面的符号，通常是一点“.”。

间距　定理头与定理内容之间的空白距离。一般是 “.5em” 表示通常两个词之间的空白。 \newline 表示定理内容另起一行。

补充　对定理头的补充说明，可以空置。此处可以用 \thmnote{#3}, 表示用方括号中的附加条目来代替定理头。

假设在源文件的导言部分有下面的命令：

```
\newtheoremstyle{break}{9pt}{9pt}{\itshape}{}{\scshape}{}{\newline}{}
\theoremstyle{break}\newtheorem{Cor}{Corollary}
\newtheoremstyle{mystyle}{3pt}{3pt}{\itshape}{\parindent}{\bfseries}{}{5mm}{}
\theoremstyle{mystyle} \newtheorem{Thm}{定理}
\newtheoremstyle{citing}{3pt}{3pt}{\itshape}{}{\bfseries}{.}{.5em}{\thmnote{#3}}
\theoremstyle{citing} \newtheorem*{citedthm}{}
```

我们就可以得到下面三个例子中的定理格式：

```
\CJKindent
下面是著名的微分中值定理：
\begin{Thm}
若~$f$~是闭区间~$[a,b]$~上的可微函数，
则存在~$t\in(a,b)$~使得
$$f(b)-f(a)=f'(t)(b-a).$$
\end{Thm}
```

下面是著名的微分中值定理：

定理 1　若 f 是闭区间 $[a,b]$ 上的可微函数，则存在 $t\in(a,b)$ 使得

$$f(b)-f(a)=f'(t)(b-a).$$

```
\begin{Cor}
This is a sentence typeset in the
theorem environment \texttt{Cor}.
\end{Cor}
```

COROLLARY 1
This is a sentence typeset in the theorem environment ***Cor****.*

```
\begin{citedthm}[Theorem 2 in \cite{tex}]
This is a sentence typeset in the
theorem environment \texttt{citedthm}.
\end{citedthm}
```

Theorem 2 in [2]. *This is a sentence typeset in the theorem environment* ***citedthm****.*

5.5.4 定理的证明环境

在 amsthm 宏包中定义了环境 proof 用来排版定理的证明。定理的证明放在这个环境中，并且这个环境会自动在证明的开始打印一个证明名称，即“Proof”，它是由命令 \proofname 来定义的。起初构造 proof 环境的目的是为了排版那些较短的不超过一页的定理证明，对于有好几页长的证明最好还是单独设为一节。

这个环境有一个选项，其中内容被用来替代证明名称，比如想要某个定理的证明由“Proof of the Main Theorem”来引导，可以输入：

```
\begin{proof}[Proof of the Main Theorem] ··· \end{proof}
```

这个环境还会在证明结束时打印一个证明结束符，即“□”。如果要想得到不同形式的证明结束符，可以用 \renewcommand 来重新定义命令 \qedsymbol。对于不使用 proof 环境的单独一小节中的证明，也可以用命令 \qed 来打印一个证明结束符。

证明结束符的位置通常在证明最后一行的右边缘。但如果最后一行是独立显示的公式，证明结束符可能会引起一些问题，此时可以在想打印证明结束符的地方使用命令 \qedhere。如果仍有问题，可以使用命令 \mbox{\qedhere}。

第 6 章　作　　图

在排版由文字组成的段落以及由段落组成的页面方面，TeX 可以说是最好的排版系统了。但是在这样一个对信息交换的需求不断增强的时代，大多数出版物都不会局限在只排版文字方面，事实上在出版物中对图形的使用已经非常普及了。然而 TeX 本身并不是一个作图系统，它本质上只是将一个一个的小盒子整齐地排列起来。不过 TeX 中也有若干命令用来画一些如点、直线、圆等简单图形。基于这些命令还出现了一些用于作图的宏包，如 curves、ebezier、epic、eepic 等，这些宏包使我们能很方便地作出一些简单的图形。在 LaTeX 中，图形都被置于 picture 环境中。本章先介绍标准 LaTeX 的 picture 环境以及各种简单的图形命令，之后再介绍一些常用的作图宏包。

6.1　作 图 环 境

6.1.1　尺寸与位置

在 LaTeX 中图形只能被放置在一个建立了坐标系的图形区域中。这个坐标区域有一个参照点（或称原点）和两条互相垂直的坐标轴，以及坐标轴上一个用来度量的单位长度。原点位于图形区域的左下角，区域的下边直线称为水平坐标轴也称 x 轴，左边直线称为垂直坐标轴或 y 轴。

一旦确定了单位长度（UL），图形区域中的每个点就都可以用一对十进制数组成的坐标来表示，点与坐标之间的对应是一对一的。坐标中的第 1 个数与 x 轴相关，称为 x 坐标，表示点在 x 轴上的投影点到原点的距离是单位长度的多少倍，第 2 个数与 y 轴相关，称为 y 坐标，表示在 y 轴上的投影点到原点的距离是单位长度的多少倍。单位长度可以用命令

```
\setlength{\unitlength}{length}
```

来确定，其中 *length* 是一个长度，如 1pt、1cm 等。图 6.1 中包含两个图形区域示例。在左边的图形中，单位长度被设置成 UL=1.2cm，因此点 (2.2,1.4) 到 x 轴的距离为 1.2cm 的1.4 倍，即 1.68cm，到 y 轴的距离为 1.2cm 的 2.2 倍，即 2.64cm。

一般情况下点的坐标是一对正数，也就是说这种点位于 x 轴的上方和 y 轴的右边。不过负数坐标也是允许的。当一点的 x 坐标的值是负数时，此点位于 y 轴的左边，而 y 坐标的值是负数时，此点就位于 x 轴的下方。图 6.1 中的右图展示了含有负数坐标的

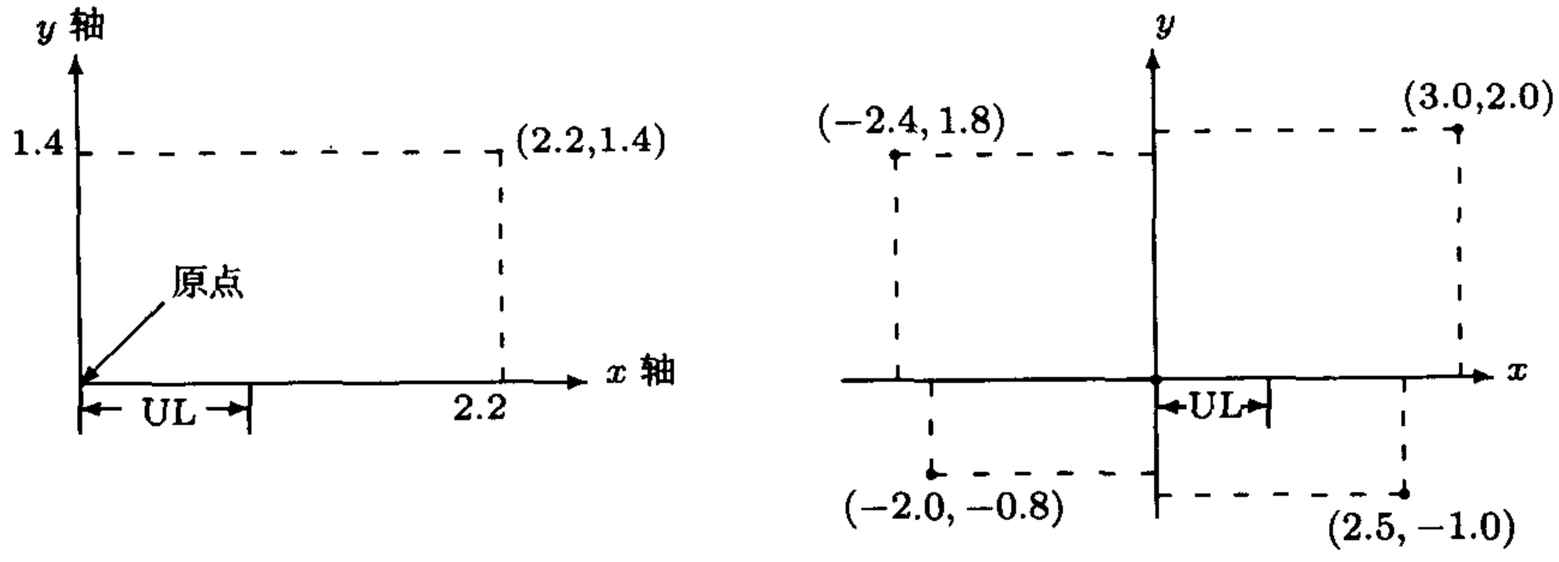


图 6.1 作图区域示例

点，此图的长度单位是 0.8cm。

为方便作图，单位长度通常设置成 1cm、1mm、1in 等。因为这样作出来的图形一旦完成，就可以根据需要很简单地通过重新设置单位长度来将图形放大或缩小到确定的倍数。比如根据单位长度 1cm 所作的图形，当单位长度重新设置为 1.5cm 时，图形就放大了 1.5 倍。

6.1.2 作图环境的语法

要创建一个图形区域，就必须使用下面的 picture 环境：

\begin{picture}(*width,height*) *picture_commands* \end{picture}

其中 *width* 代表所创建的图形的宽度， *height* 代表所创建的图形的高度。这里的宽度和高度都是十进制数，它们分别表示实际宽度和高度是单位长度的多少倍。通常在使用 picture 环境之前，都先设置单位长度。比如下面的命令

```
\setlength{\unitlength}{1.5cm}
\begin{picture}(4,5) … \end{picture}
```

创建了一个单位长度是 1.5cm 的图形区域，其宽度是单位长度的 4 倍，高度是单位长度的 5 倍。如果在 picture 环境之前没有设置单位长度，则单位长度取默认值 1pt。

picture 环境中的 *picture_commands* 是一些产生或放置图形元素的命令。所谓图形元素就是点、直线段、有向直线段、圆等基本图形。

要注意在 picture 环境中不能再改变单位长度，整个环境中的单位长度必须是统一的。不过嵌套在外层作图环境中的子作图环境可以有不同的单位长度。

6.1.3 放置图形元素

图形区域中的图形由各个图形元素组成，在下一小节中将详细介绍如何生成各种图形元素。在标准 LaTeX 中，每个图形元素都由命令 \put 或 \multiput 放入图形区域中。这两条命令的语法是：

\put(*x_coord,y_coord*){*pic_elem*}

\multiput(*x_coord,y_coord*)(*x_incr,y_incr*){*num*}{*pic_elem*}

其中 *pic_elem* 是某个图形元素；*x_coord* 和 *y_coord* 分别是点在图形区域坐标系中的 x 坐标和 y 坐标，此点也就是 *pic_elem* 所处的位置。事实上，命令 \put 以 (*x_coord,y_coord*) 为参照点建立一个盒子，然后将 *pic_elem* 放置其中。

命令 \multiput 将同一个图形元素以 (*x_coord,y_coord*) 为起点放置 *num* 次（*num* 是一个自然数），每放置一次，其位置的 x 坐标和 y 坐标就分别以 *x_incr* 和 *y_incr* 为增量移动到下一个坐标。因此，上面的 \multiput 命令就将 *pic_elem* 连续地放入下面的 *num* 个位置上：

(*x_coord*, *y_coord*), (*x_coord*+*x_incr*, *y_coord*+*y_incr*),

(*x_coord*+2*x_incr*, *y_coord*+2*y_incr*), $\cdots$

(*x_coord*+[*num*−1]*x_incr*, *y_coord*+[*num*−1]*y_incr*)

增量 *x_incr* 和 *y_incr* 都是十进制数，可以是正数也可以是负数。因而命令

```
\multiput(2.5,3.6)(0.5,-0.6){5}{pic_elem}
```

将图形元素 pic_elem 放置 5 次，第 1 次放在点 (2.5,3.6)，然后依次放在点 (3.0,3.0)、点 (3.5,2.4)、点 (4.0,1.8)，最后放在点 (4.5,1.2)。

注意，坐标及增量必须用圆括号括起来，次数及图形元素都要用花括号括起来。另外，分隔 x 坐标和 y 坐标必须用逗号，小数点必须用英文句号，即一个黑点。

6.2 图 形 元 素

6.2.1 文本

最简单的图形元素是由纯文本组成的，也就是说在上节所介绍的命令 \put 和 \multiput 中的 *pic_elem* 可以是任何形式的文本。例如，命令

```
\put(2.5,3.6){An arrow}
```

把文本 “An arrow” 以坐标 (2.5,3.6) 为参照点插入到图形区域中，如图 6.2 所示。

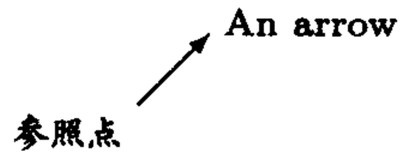


图 6.2 文本元素

文本也可以先用 \parbox 命令或 minipage 环境包装起来，然后再放入图形区域中。此时，\parbox 命令和 minipage 环境的参照点就是放置位置的坐标点。例如：

\parbox[b]{···}{···}
以 b 为位置选项的
段落盒子的参照点
在左下角

\parbox{···}{···}
无选项的段落盒子
的参照点在左边缘
的中间

\parbox[t]{···}{···}
以 t 为位置选项的
段落盒子的参照点
在左上角

其中的黑点代表 \parbox 的参照点。

6.2.2 文本的垂直叠放

在 3.14.6 节中所介绍的命令 \shortstack 也可以用在作图环境中当作图形元素来使用，其作用是将较短的文本垂直叠放。事实上，此命令会先建立一个盒子，然后将文本一行一行叠放在盒子中。盒子的左下角位置就是 \put 指明的坐标点。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\put(10,5){\shortstack[l]{This\\
  is\\the\\left\\column}}
\put(30,5){\shortstack{This\\
  is\\the\\middle\\column}}
\put(50,5){\shortstack[r]{This\\
  is\\the\\right\\column}}
\end{picture}
```

This is the left column　This is the middle column　This is the right column

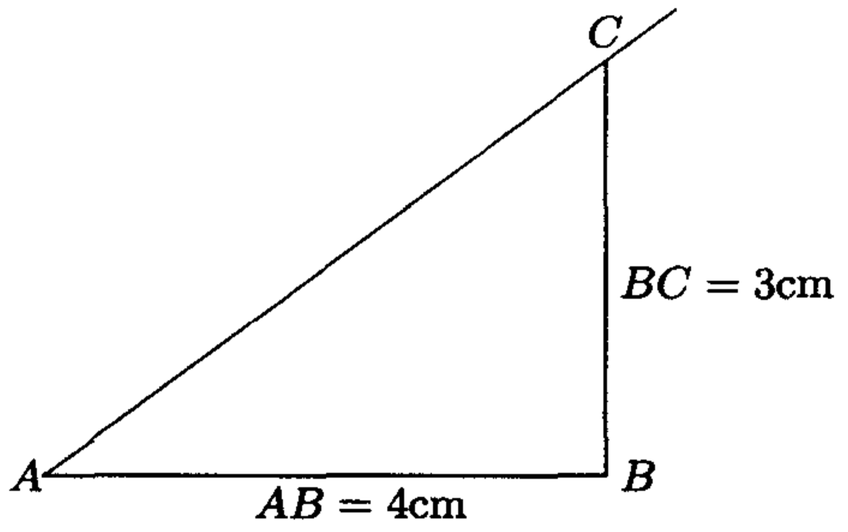
6.2.3 直线段

在作图环境中，可以画有任意长度的水平和垂直线段以及各种斜率满足一定条件的倾斜直线段。画直线段的语法是：

\line($\Delta x, \Delta y$){*length*}

其中 *length* 是长度参数。对于水平直线段和垂直线段来说，这里的 *length* 就代表所画直线段的实际长度是单位长度的多少倍。但是对于倾斜直线段，*length* 的意义就有些复杂，它并不恰好代表所画直线段的实际长度，而是代表这条倾斜直线段在水平方向上的投影线段的长度。$(\Delta x, \Delta y)$ 是斜率对，它决定了所画直线段的斜率。当 $\Delta x = 1$, $\Delta y = 0$ 时，所画线段是指向右边的水平线段；当 $\Delta x = 0$, $\Delta y = 1$ 时，所画线段是指向上方的垂直线段。一般情况下，斜线的斜率是比值 $\Delta y/\Delta x$。

```
\setlength{\unitlength}{1cm}
\begin{picture}(6,4)
\put(1,0.5){\line(1,0){4}}
\put(5,0.5){\line(0,1){3}}
\put(1,0.5){\line(4,3){4.5}}
\put(1,0.5){\makebox(0,0)[r]{$A$}}
\put(5.1,0.5){\makebox(0,0)[l]{$B$}}
\put(5,3.6){\makebox(0,0)[b]{$C$}}
\put(2.5,0.2){$AB=4$cm}
\put(5.1,1.8){$BC=3$cm}
\end{picture}
```



在上面这个例子中，我们先从 A 到 B 画了一条 4cm 长的水平线段，然后从 B 到 C 画了一条 3cm 长的垂直线段，最后从 A 沿着斜率为 3/4 的方向画了一条长度参数为 5 的斜线段。我们看到斜线段的长度实际要比 AC 的长度更长，因为它在 x 轴上的投影的长度是 5cm。如果把斜线的长度参数改为 4，那么这三条线段就正好形成一个直角三角形了。

如前所述，标准 LaTeX 只能画一些斜率满足一定条件的倾斜直线段。事实上，斜率对中的 Δx 和 Δy 必须满足下面三个条件：

1) Δx 和 Δy 的值只能取整数（可以是负数）。

2) 这些整数的绝对值只能是 $0, 1, \cdots, 6$ 之一。

3) Δx 和 Δy 不能有大于 1 的公约数。

按照这几条规则，(3.5,1.5) 和 (7,0) 不能作为斜率对。(2,2) 和 (3,6) 因不满足第三条规则，也不能作为斜率对，其实它们与 (1,1) 和 (1,2) 分别对应着同样的斜率。事实上，满足这几条规则且 Δx 和 Δy 不是负数的斜率对只有 25 个。另外，Δx 和 Δy 也可以是负数。Δx 为负数表示直线是从右向左画的，Δy 为负数则表示直线是从上往下画的。

注意，倾斜直线段的长度不能小于 10pt 或者 3.5mm，否则什么也画不出来，这是因为 LaTeX 实际上是用一些特殊的小线段来产生倾斜直线段的，而这些小线段的长度的最小值不小于 10pt 或者 3.5mm。

6.2.4 有向直线段

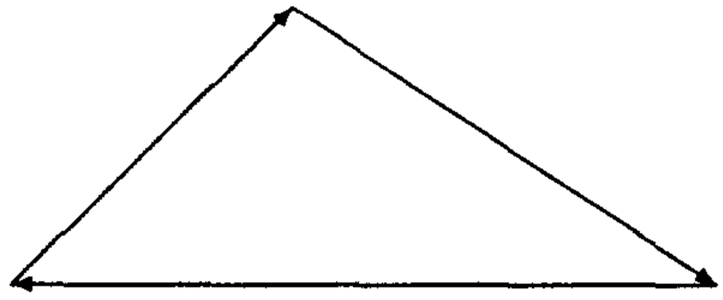
有向直线段就是在线段的一端带有箭头的直线段，它可以用下面的命令

\vector($\Delta x, \Delta y$){*length*}

画出来，其中 $(\Delta x, \Delta y)$ 是斜率对，*length* 是有向直线段的长度参数。命令 \vector 的使用方法与 \line 命令的使用方法一样。在作图环境中，\vector 命令以 \put 命令或

\multiput 命令所指明的坐标点为起点，按照斜率对所对应的方向，画一条或多条直线段，并在终点处画上指示方向的小箭头。\vector 命令也要满足 \line 命令的所有限制条件，而且斜率对中的 Δx 和 Δy 的绝对值只能取 $0, 1, \cdots, 4$ 这几个数。因而当 Δx 和 Δy 不是负数时，有向直线段的方向总共只有 13 个。下面是一个例子：

```
\setlength{\unitlength}{1cm}
\begin{picture}(5,2)
\put(5,0){\vector(-1,0){5}}
\put(0,0){\vector(1,1){2}}
\put(2,2){\vector(3,-2){3}}
\end{picture}
```



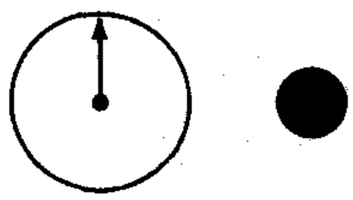
6.2.5 圆

在作图环境中，可以用下面两条命令

```
\circle{diameter}
\circle*{diameter}
```

来画圆，其中 *diameter* 是圆的直径参数，实际直径等于单位长度乘以直径参数。上面的第一条命令以 \put 或者 \multiput 指明的坐标为圆心，以 *diameter* 为直径参数画一个或多个圆。第二条带星号命令的用法与第一条一样，但所画的圆是实心的。注意，在标准 LaTeX 中实际上只有有限个值供圆的直径使用。系统实际上是从这有限个值中挑选一个最接近所给数的值当作圆的直径。当所给直径较大（比如大于 15mm）时，并不能得到一个直径恰好是所给值的圆。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,16)
\put(10,10){\circle*{1}}
\put(10,10){\circle{12}}
\put(10,10){\vector(0,1){6}}
\put(25,10){\circle*{5}}
\end{picture}
```



6.2.6 圆角矩形

长方形的四个直角被替换成四分之一圆弧后所形成的图形称为圆角矩形。圆弧的直径取决于长方形的长和宽，且直径的选择总是使得在衔接处尽量光滑。圆角矩形可以用下面的命令得到：

\oval(*width*,*height*)[*portion*]

其中 *width* 和 *height* 分别是形成圆角矩形的那个长方形的长和宽相对于单位长度的倍数。圆角矩形的中心就是由 \put 或 \multiput 所指明的坐标点，如图 6.3 所示。选项 *portion* 用来指明要画出的是圆角矩形的哪个部分，它可以取字母 t、b、l 或 r 以及它们的一些组合。下面是各种选项的意义：

t	画上半个圆角矩形。
b	画下半个圆角矩形。
l	画左边半个圆角矩形。
r	画右边半个圆角矩形。
tl	画左上方的四分之一圆角矩形。
tr	画右上方的四分之一圆角矩形。
bl	画左下方的四分之一圆角矩形。
br	画右下方的四分之一圆角矩形。

```
\put(35,5){\oval(40,15)}
```

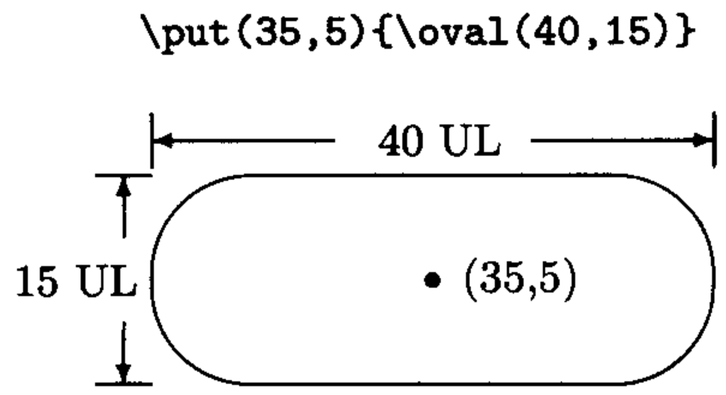
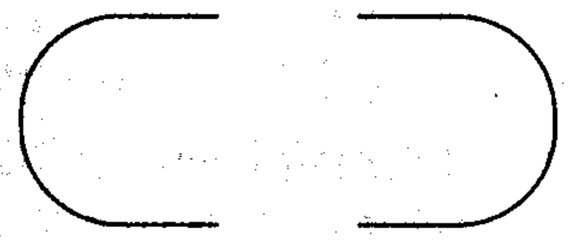


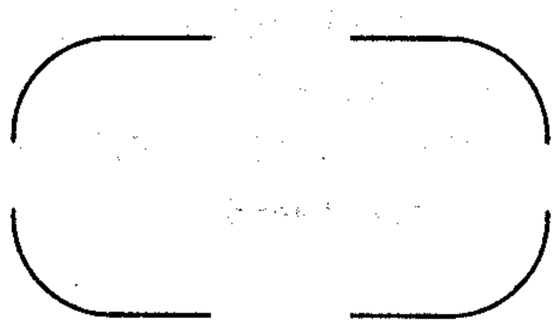
图 6.3 圆角矩形示例

在有两个字母的选项中，字母组合的顺序是无关紧要的，因而 lt、rt、lb 和 rb 也是有效的选项，它们分别等同于上面列出的后面 4 个选项。下面的两个例子展示了圆角矩形的各个部分：

```
\setlength{\unitlength}{1mm}
\begin{picture}(40,20)
\put(30,10){\oval(28,15)[l]}
\put(40,10){\oval(28,15)[r]}
\end{picture}
```



```
\setlength{\unitlength}{1mm}
\begin{picture}(40,25)
\put(30,15){\oval(28,15)[lt]}
\put(40,15){\oval(28,15)[rt]}
\put(30,10){\oval(28,15)[lb]}
\put(40,10){\oval(28,15)[rb]}
\end{picture}
```



6.2.7 Bézier 曲线

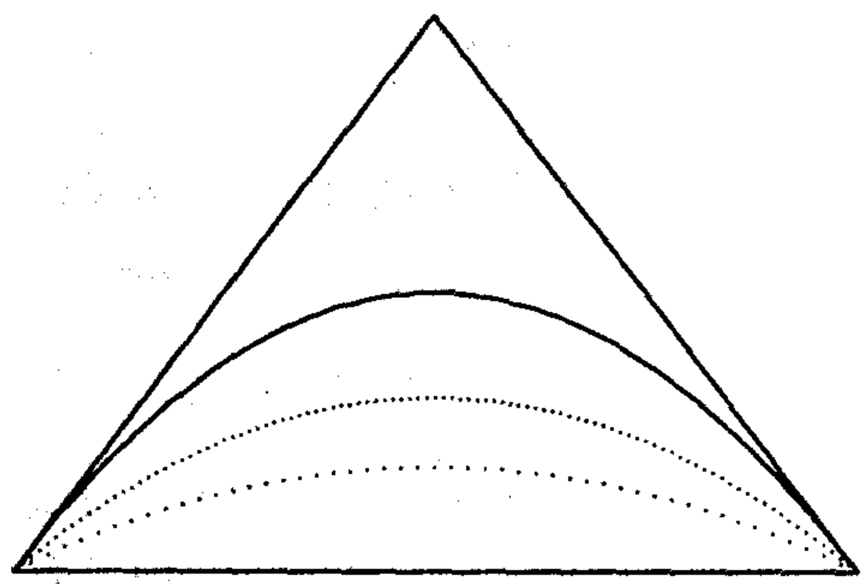
Bézier 曲线是由系数为一些坐标点的 Bernstein 多项式构造的。这些坐标点被称为控制点。连接最前面两个控制点以及连接最后面两个控制点得到的两个线段会与

Bézier 曲线分别在起点和终点处相切。$n+1$ 个控制点可以决定一条 n 次 Bézier 曲线。在 LaTeX2e 的作图环境中可以使用下面的两条命令

```
\qbezier[n](Ax,Ay)(Bx,By)(Cx,Cy)
\bezier{n}(Ax,Ay)(Bx,By)(Cx,Cy)
```

来得到一条二次 Bézier 曲线。上面第一条命令中的 (Ax,Ay)、(Bx,By) 和 (Cx,Cy) 就是决定这条二次 Bézier 曲线的 3 个控制点的坐标。可选项 n 是一个自然数，它指示 LaTeX 用多少个点来描绘 Bézier 曲线。当 3 个控制点在同一条直线上时（如当 $Bx=(Ax+Cx)/2$, $By=(Ay+Cy)/2$ 时），所得到的 Bézier 曲线恰好就是连接首尾两个控制点的直线段。利用这个性质可以得到有任意斜率的直线段。下面我们先用 3 个在同一直线上的控制点画一条水平直线，紧接着画 3 条 Bézier 曲线，最后画出在首尾两点与 Bézier 曲线相切的两条线段。

```
\setlength{\unitlength}{1cm}
\begin{picture}(6,4)
\qbezier(0,0)(3,0)(6,0)
\qbezier[50](0,0)(3,1.5)(6,0)
\qbezier[100](0,0)(3,2.5)(6,0)
\qbezier(0,0)(3,4)(6,0)
\qbezier(0,0)(1.5,2)(3,4)
\qbezier(3,4)(4.5,2)(6,0)
\end{picture}
```



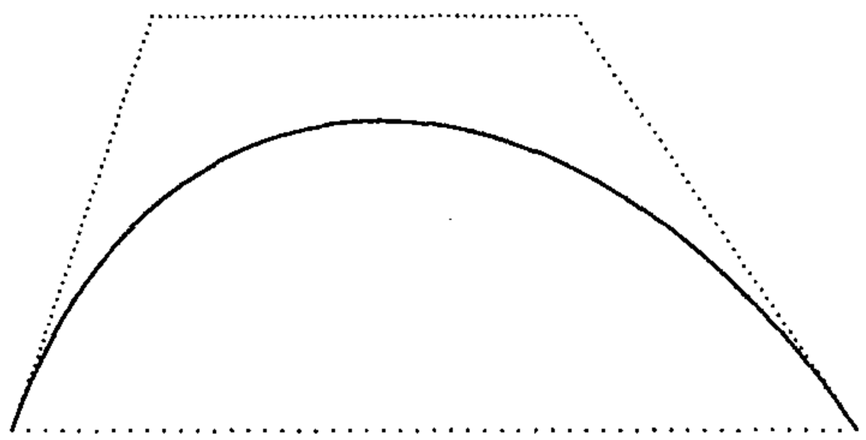
第二条命令与第一条命令的作用相同，只是其中的 n 是必需的参数，并非可选项。之所以保留这条命令是为了兼容 bezier 宏包中的 \bezier 命令。

ebezier 宏包扩充了 LaTeX2e 画 Bézier 曲线的功能，它能用四个控制点画三次 Bézier 曲线，以及用两个控制点画一次 Bézier 曲线（即：直线）。下面是 ebezier 宏包提供的两条命令：

```
\lbezier[n](Ax,Ay)(Bx,By)
\cbezier[n](Ax,Ay)(Bx,By)(Cx,Cy)(Dx,Dy)
\Cbezier[n](Ax,Ay)(Bx,By)(Cx,Cy)(Dx,Dy)
```

其中第一条命令用来画连接两点的直线，第二条命令用来画有 4 个控制点的三次 Bézier 曲线，第三条命令除了画出 Bézier 曲线外还将控制点以及连接控制点的直线画出。

```
\setlength{\unitlength}{1cm}
\begin{picture}(6,3)
\cbezier[500](0,0)(1,3)(4,3)(6,0)
\lbezier[50](0,0)(1,3)
\lbezier[50](1,3)(4,3)
\lbezier[50](4,3)(6,0)
\lbezier[50](0,0)(6,0)
\end{picture}
```



ebezier 宏包还定义了画圆和圆弧的命令，这里不再叙述。

6.2.8 盒子

盒子命令 \frambox、\makebox 和 \savebox 也可以作为图形元素在作图环境中使用，但在作图环境中使用这些盒子命令时，它们的语法与在 2.14 节中所介绍的稍有不同。另外，在作图环境中还可以用 \dashbox 命令来产生有虚线外框的盒子。

\makebox(*width,height*)[*pos*]{*text*}

\framebox(*width,height*)[*pos*]{*text*}

\dashbox{*dash_length*}(*width,height*)[*pos*]{*text*}

在上面的三条命令中 *width* 和 *height* 是一对十进制数，分别表示盒子的宽度和高度是单位长度的多少倍；*pos* 是位置选项，表示盒子中的文本在盒子中所处的位置。位置选项可以取 t、b、l、r 这 4 个字母以及它们的一些组合，其意义说明如下：

t	盒子中的文本位于盒子的顶部居中。
b	盒子中的文本位于盒子的底部居中。
l	盒子中的文本位于盒子的左边并垂直居中。
r	盒子中的文本位于盒子的右边并垂直居中。
tl	盒子中的文本位于盒子顶部的左边。
tr	盒子中的文本位于盒子顶部的右边。
bl	盒子中的文本位于盒子底部的左边。
bl	盒子中的文本位于盒子底部的右边。

```
\put(20,15){%
  \framebox(40,15){ 正中间}}
```

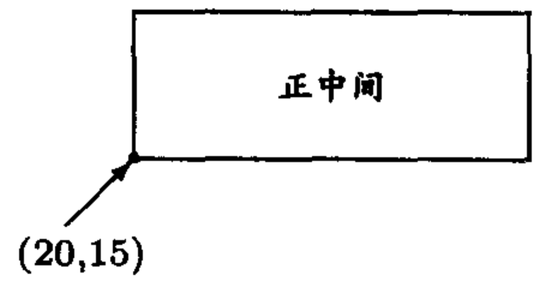
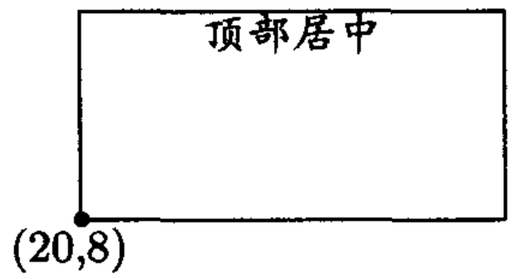


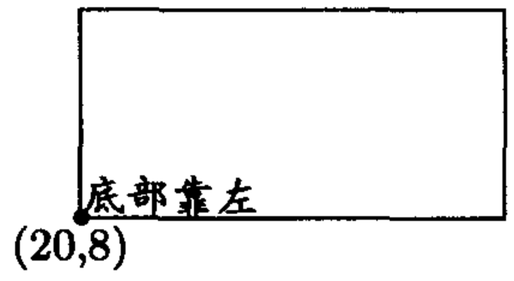
图 6.4 盒子图形元素

位置选项中字母组合的顺序是无关紧要的，因而 tl 与 lt 的结果一样。当位置选项省略时，文本居于盒子的正中间，如图 6.4 所示，其中所用的单位长度是 0.7mm。下面以命令 \framebox 为例，观察位置选项对文本在盒子中的位置的影响：

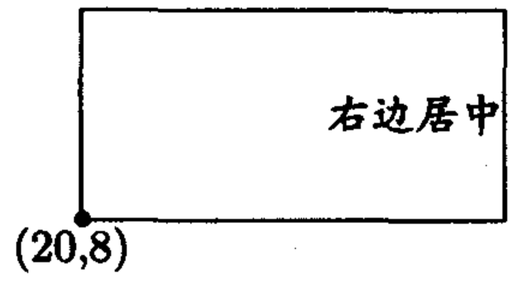
```
\setlength{\unitlength}{1mm}
\begin{picture}(50,25)
\put(20,8){\framebox(30,15)[t]{%
\small\itshape 顶部居中}}
\put(20,8){\circle*{1}}
\put(15,5){(20,8)}
\end{picture}
```




```
\setlength{\unitlength}{1mm}
\begin{picture}(50,25)
\put(20,8){\framebox(30,15)[lb]{%
\small\itshape 底部靠左}}
\put(20,8){\circle*{1}}
\put(15,5){(20,8)}
\end{picture}
```




```
\setlength{\unitlength}{1mm}
\begin{picture}(50,25)
\put(20,8){\framebox(30,15)[r]{%
\small\itshape 右边居中}}
\put(20,8){\circle*{1}}
\put(15,5){(20,8)}
\end{picture}
```



图形元素 \makebox 的使用方法与 \framebox 一样，只是盒子没有边框线。这里需要特别指出的是，由 \makebox 构造的宽度和高度都是 0 的盒子经常用来将文本放在一个特别的位置。从下面四个零宽度和零高度的 \makebox 命令的输出效果，可以看出盒子中文本位置的变化，这里已经把单位长度设置为 1mm。

```
\put(30,16){\makebox(0,0){盒子中间}}
\put(20,5){\makebox(0,0)[tr]{顶部靠右}}
\put(40,10){\makebox(0,0)[b]{底部居中}}
\put(20,28){\makebox(0,0)[l]{左边居中}}
```

图 6.5 中箭头所指之处即为相应 \makebox 盒子的参照点。另外选项为 lb 时的输出效果与不使用 \makebox 而直接简单地把文本作为图形元素时的效果一样。

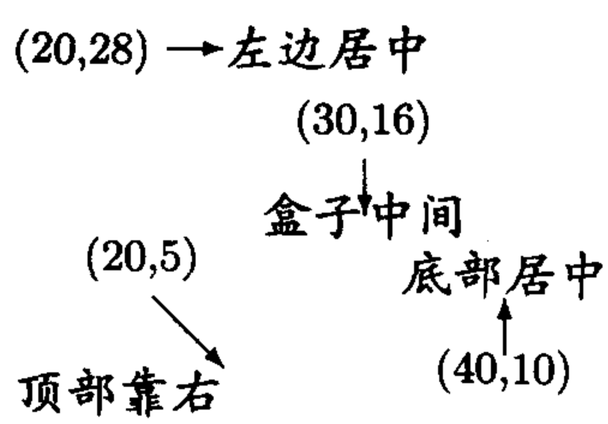
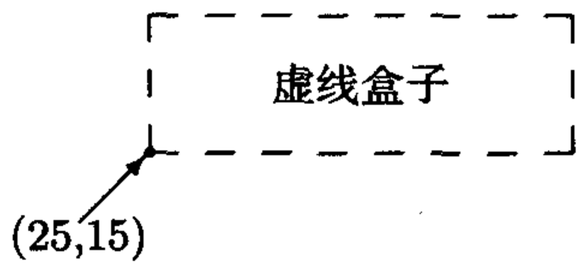


图 6.5 \makebox 图形元素

图形元素 \dashbox 也产生一个带边框的盒子，但边框线是用虚线构成的，虚线中每一小段的长度由 \dashbox 的第一个参数 *dash_length* 确定。*dash_length* 也是一个十

进制数，表示虚线中每一小段的长度是单位长度的多少倍。只有当 *dash_length* 是盒子宽度的倍数，同时也是高度的倍数时，虚线盒子的输出效果才显得最好。下面是一个虚线盒子的例子：

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,30)
\put(25,15){\dashbox{2}(30,10){虚线盒子}}
\put(25,15){\circle*{0.7}}
\put(20,10){\vector(1,1){4.7}}
\put(15,8){\small(25,15)}
\end{picture}
```



各种盒子图形元素中的文本仍可以事先放入一个如 \parbox 或 minipage 的盒子中。不过这些段落盒子本身也有位置选项 t 或 b。因此，处于内层的这些盒子的位置选项不应该与处于外层的盒子图形元素的位置选项相冲突，它们应遵循如下法则：

> 如果盒子图形元素中的位置选项中含有 t 或 b，那么内层的段落盒子也必须有同样的位置选项。如果盒子图形元素不含位置选项或选项中只含有 r 或 l，那么内层的段落盒子只能是标准形式，即不含位置选项。

6.2.9 显示文本框

在作图环境中可以使用命令

\frame{文本内容}

来将文本内容的边框显示出来。边框与文本内容之间没有额外的空隙。这里的文本内容可以是其他的盒子。事实上这条命令相当于带边框的 \mbox 命令。另外这条命令也可以在作图环境之外使用。例如：

```
\setlength{\unitlength}{1cm}
\begin{picture}(5,3)
\put(5,0.5){\frame{%
\shortstack{W\\O\\R\\D}}}
\put(1,0.5){\frame{%
\makebox[2cm][r]{TEXT}}}
\end{picture}
```

6.3 其 他 命 令

6.3.1 线条的粗细

在作图环境中我们可以用命令

\thicklines 和 \thinlines

来改变圆、圆角矩形、有向直线段和倾斜直线段的线条粗细。每条命令的作用会持续下去直到另一条命令出现为止。一进入作图环境 LaTeX 就自动调用命令 \thinlines。对于水平直线段和垂直直线段，可以用命令

\linethickness{*thickness*}

来改变线条的粗细，其中参数 *thickness* 是一个长度，表示粗细程度，如 0.5pt、1mm 等。这条命令也会持续作用下去直到另一条这样的命令出现。

对于带边框的盒子如 \framebox 和 \dashbox, 由于其边框是由水平直线和垂直直线构成，所以命令 \linethickness 也会对它们起作用。

6.3.2 图形的嵌套

作图环境本身也可以作为一个图形元素放置到另一个作图环境中，也就是说作图环境可以嵌套使用。一般来说是用下面的语法

```
\put(x_coord,y_coord){\setlength{\unitlength}{UL}
\begin{picture}(width,height)
子图形命令
\end{picture}}
```

来放置一个子图形。子作图环境的坐标系原点就是外层环境中 \put 命令指明的坐标点。子作图环境坐标系的单位长度可以不同于外层环境的单位长度，但在没有设置子图形的单位长度时，子图形的单位长度就与外层环境的单位长度一样。

6.3.3 图形的存储和提取

一组图形元素可以事先储存在一个盒子中，然后通过反复调用整个盒子来多次使用这些图形元素，而不需要每次都键入这些图形元素命令。要做到这一点，我们必须先使用命令

\newsavebox{*sub_pic_name*}

来预订一个名为 *sub_pic_name* 的盒子。注意，这个名字必须是标准命令的形式。然后再使用命令

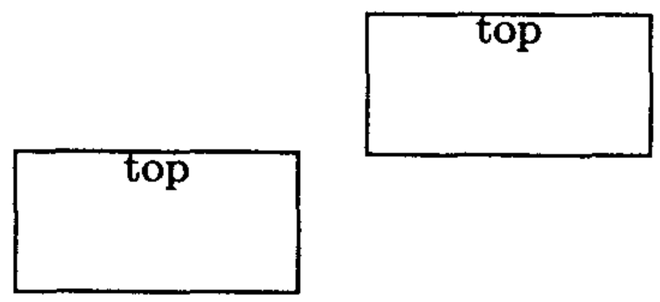
\savebox{*sub_pic_name*}(*width*,*height*)[*pos*]{*sub_pic*}

将子图形储存在这个盒子中，其中 *sub_pic* 就是组成子图形的命令，而 *width* 和 *height* 分别是装子图形的盒子的宽度和高度，*pos* 是位置选项，其使用方法与 6.2.8 节中 \makebox 盒子元素一样。最后在作图环境中使用命令

\usebox{*sub_pic_name*}

把子图形当作一个新的图形元素，并用 \put 或者 \multiput 命令把它放置到作图环境中。下面是一个图形储存与提取的例子：

```
\setlength{\unitlength}{1cm}
\newsavebox{\sub}
\savebox{\sub}(2,1)[t]{top}
\begin{picture}(4,3)
\put(1.5,0){\frame{\usebox{\sub}}}
\put(4,1){\frame{\usebox{\sub}}}
\end{picture}
```



6.3.4 图形的位移

如果需要将已经设计好的图形做适当的位移，那么可以用下面扩展的作图环境

\begin{picture}(*width*,*height*)(*x_offset*,*y_offset*)
图形命令
\end{picture}

与 6.1.2 节的标准作图环境相比，这里多了 *x_offset* 和 *y_offset*, 它们分别表示图形的水平方向位移量和垂直方向位移量，这两个位移量仍然是单位长度的倍数。当 *x_offset* 为正数时，整个图形向左移动，反之，则向右移动；当 *y_offset* 为正数时，整个图形向下移动，反之，则向上移动。

6.4 作图环境的功能扩展

在前面几节中我们已经阐述了如何利用标准 LaTeX 的作图环境来画一些基本的图形。由于作图功能的限制，能够画的图形还是很有限的。比如要画一条光滑曲线经过给

定的不在同一直线上的三点，就很难做到。另外作一幅图形的工作量也很大，大多数画图命令都需要明确给出各个图形元素的坐标。在这一节中我们介绍几个常用作图宏包，它们拓展了 LaTeX 作图环境的功能，使我们能比较轻松地画出更多更复杂的图形。

6.4.1 基本作图环境的扩展 —— epic 宏包

作为标准 LaTeX 作图环境的扩展，epic 宏包提供了如下的新命令：

```
\multiputlist   \jput      \matrixput   \dottedline   \dashline
\picsquare      \grid      \drawline    \putfile
```

以及在作图环境中使用的三个环境:

`dottedjoin` 环境　　`dashjoin` 环境　　`drawjoin` 环境

下面逐个说明它们的意义和使用方法。

\multiputlist$(x,y)(\Delta x,\Delta y)$[pos]{$item1$,$item2$,$\cdots$,$itemN$}

这条命令是标准 LaTeX 中 \multiput 命令的变形。它允许我们在规则分布的点上放置各种不同的图形元素。命令中的 (x,y) 是起始坐标，Δx 和 Δy 分别是坐标在水平方向上和垂直方向上的增量。$item1$,$item2$,$\cdots$,$itemN$ 是各项图形元素，它们之间用逗号分开，不能用空格。如果单个的图形元素中含有逗号，则这个图形元素就要用一对花括号括起来。如果一行写不下，则要在行尾使用百分号 "%" 消去换行时出现的空格。命令中的 pos 是位置选项，它可以取 t、b、r、l 这些值。实际上，这条命令为每个项目使用 \makebox(0,0)[pos]{$\cdots$} 建立一个盒子，然后将这个项目放入盒子中。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,20)
\multiputlist(20,2)(10,5){%
This,is,a,example}
\end{picture}
```

example
a
is
This

随着坐标位置的移动，命令从所列出的图形元素中逐个提取项目，并按顺序放在坐标点上。第一个项目放在第一个坐标点，第二个项目放在第二个坐标点，直到项目被放完。

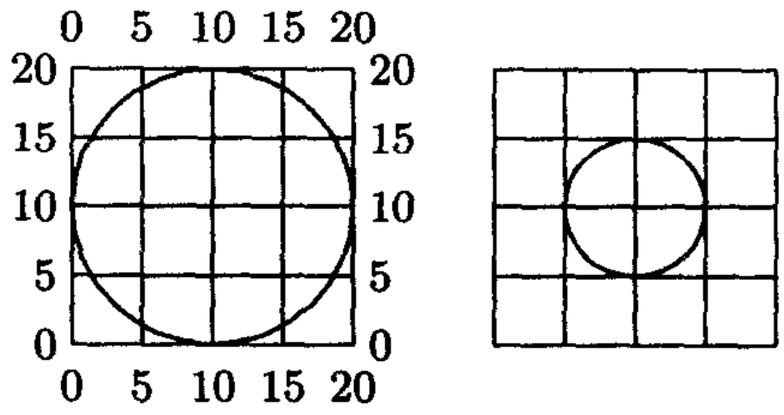
下面的命令：

\grid$(width,height)(\Delta width,\Delta height)$[$ini\_x$,$ini\_y$]

创建一个坐标网，其中 $width$ 和 $height$ 分别是坐标网的宽度参数和高度参数，实际宽度和高度分别是宽度参数和高度参数乘以单位长度。$\Delta width$ 和 $\Delta height$ 分别是每一个小格子的宽度参数和高度参数，它们的值必须能分别整除 $width$ 和 $height$ 的值，否则将画

不出坐标网。选项 *ini_x* 和 *ini_y* 分别是在水平方向上和垂直方向上的坐标起点。如果不给出选项的值，则只画出网格，而不在网格边缘标出坐标。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\put(10,5){\grid(20,20)(5,5)[0,0]}
\put(40,5){\grid(20,20)(5,5)}
\put(20,15){\bigcircle{20}}
\put(50,15){\bigcircle{10}}
\end{picture}
```

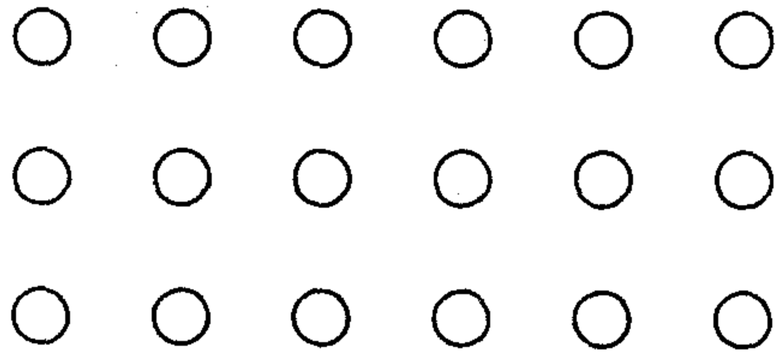


这里使用了 curves 宏包中的 \bigcircle 命令来画直径较大的圆。

\matrixput$(x,y)(\Delta x_1,\Delta y_1)\{n_1\}(\Delta x_2,\Delta y_2)\{n_2\}\{object\}$

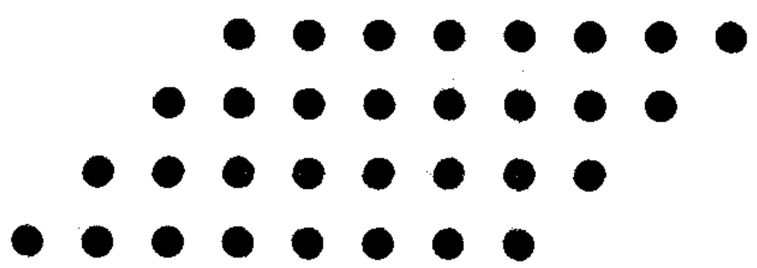
这条命令也是标准 LaTeX 中 \multiput 命令的变形。它允许我们在二维规则分布的坐标点上放置同一样东西。其中 (x,y) 是起始坐标，$(\Delta x_1,\Delta y_1)$ 是在第一维上的坐标增量，n_1 是在这一维上放置 *object* 的次数；$(\Delta x_2,\Delta y_2)$ 是在第二维上的坐标增量，n_2 是相应于这一维上放置 *object* 的次数。因此 *object* 总共被放置了 $n_1 \times n_2$ 次。例如，下面的命令将一个圆放置了 $6 \times 3 = 18$ 次：

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,25)
\matrixput(10,2)(10,0){6}(0,10){3}{%
\circle{4}}
\end{picture}
```



再比如下面的命令将一个实心圆放置了 $8 \times 4 = 32$ 次。由于在第二维的 x 方向有非零的增量，所以整个布局形成平行四边形。

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,20)
\matrixput(10,2)(5,0){8}(5,5){4}{%
\circle*{2}}
\end{picture}
```

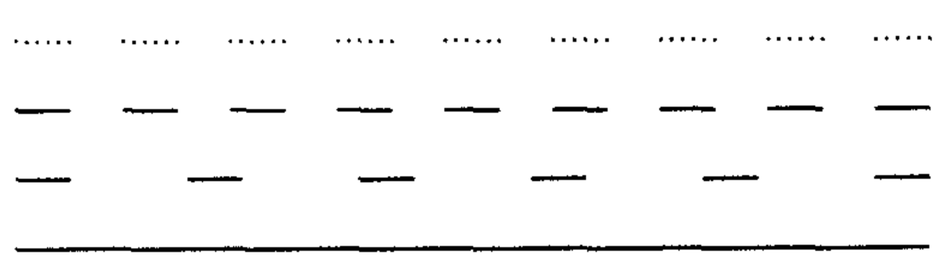


\dashline[*stretch*]{*dash_length*}[*inter_dot_gap*]$(x_1,y_1)(x_2,y_2)\cdots(x_n,y_n)$

这条命令输出一条由小线段组成的连接所给坐标点的虚线。每个小线段都是由 \dottedline 构造的。小线段的长度由参数 *dash_length* 决定，它表示单位长度的倍数。选项 *inter_dot_gap* 是构造小线段的相邻两个小点之间的空隙，它也是单位长度的倍数。

在默认的情况下，小线段看起来是实线段，但其外观可由选项 *inter_dot_gap* 的值来改变。*inter_dot_gap* 的值大于 0.4mm 时，小线段看起来就是由点组成的虚线段。下面的例子展示了几条由 \dashline 命令画的线：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,20)
\dashline{4}[0.7](0,18)(65,18)
\dashline{4}(0,13)(65,13)
\dashline[-30]{4}(0,8)(65,8)
\dashline[100]{4}(0,3)(65,3)
\end{picture}
```



选项 *stretch* 的默认值是 0，表示伸缩度。此时系统会用尽可能少的小线段来画虚线，并同时保证小线段的总长度与空白的总长度尽可能相等。当 *stretch* 为正数时，小线段的个数就被增加了百分之 *stretch*；反之，当 *stretch* 为负数时，小线段的个数就减少了相同的百分比。*stretch* 的值不能小于 −100。虽然它的值没有上界，但最好不要超过 100。这个值大于或等于 100 时，所画的已经不是虚线而是实线了，如上面例子中的最后一条线。

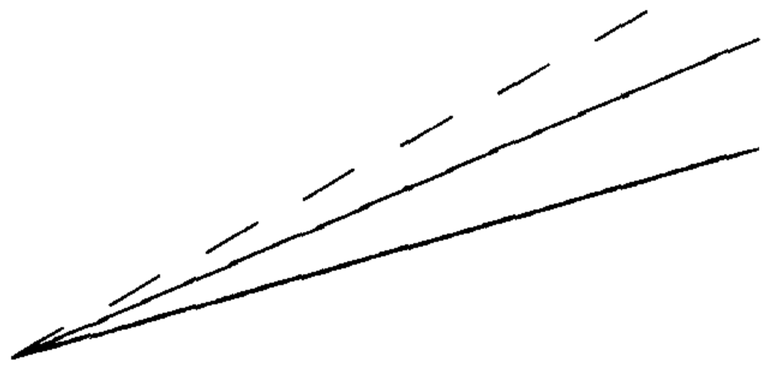
下面这条命令：

\drawline[*stretch*]$(x_1,y_1)(x_2,y_2)\cdots(x_n,y_n)$

表示用直线段将所给坐标点按顺序连接起来，因而至少应给出两个坐标点。由于 LaTeX 是用非常小的有限个小线段来构造直线段的，而这些小线段只有有限个斜率，所以这里构造出来的某些直线段看起来可能不是很平滑。可以用命令 \thinlines 将 \drawline 画出来的线设置为细线，也可以用命令 \thicklines 将其设置为粗线。

选项 *stretch* 的意义与命令 \dashline 中的 *stretch* 相似。它的默认值也是 0，不过此时这个默认值意味着要用尽量少的小线段将坐标点连接起来，并且使相邻两个小线段的首尾“相连”，从而使整个线看起来是实线。将选项 *stretch* 的值设置为负数就会使所画出来的线呈虚线。反之，将其值设置为正数表示要画更多的小线段，这样就减少了不平滑的现象，同时也使画出来的线显得黑粗些。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,25)
\drawline[-50](0,0)(45,25)
\drawline(0,0)(53,23)
\drawline[100](0,0)(53,15)
\end{picture}
```

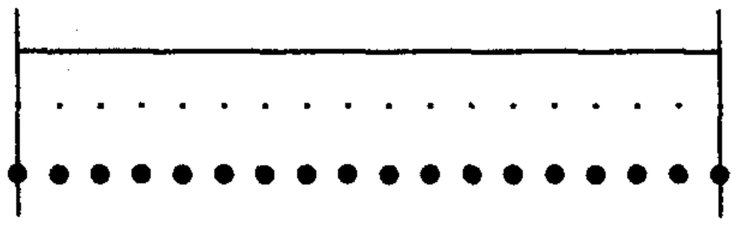


上面例子中的第一条线由于使用了负值选项画出的线是虚线。由于第二条倾斜线的斜率23/53不在LaTeX的小线段库的斜率范围之内，所以这条线看起来不是很平滑。画第三条线时，如果不使用选项值100，则画出的线也没有这么平滑。

\dottedline[*dot_character*]{*dot_gap*}$(x_1,y_1)(x_2,y_2)\cdots(x_n,y_n)$

这条命令能画出由点组成的虚线。其中参数 *dot_gap* 表示相邻两个小点之间的距离是单位长度的多少倍。选项 *dot_character* 指示用何种符号作为小点，在默认的情况下，是用小方块作为小点的。(x_1,y_1), (x_2,y_2),$\cdots$, (x_n,y_n) 是n个坐标点。由于这条命令是用小点将每两个相邻的坐标点连接起来，因此，最少应给出两个坐标点。由于画出的小点的个数必定是正整数，所以相邻两个小点之间的距离一般不能恰好等于单位长度的 *dot_gap* 倍，不过也是非常接近这个数的。参数 *dot_gap* 也可以省略。当小点是默认的符号时，虚线的粗细可以由命令 \thinlines、\thicklines 或者 \linethickness 来改变。除了某些符号（如罗马字体中的星号 *）表示的小点之外，所有“小点”都是居中摆放的，所以当“小点”的尺寸较大时，虚线可能在两边超过边界。于是一般都选择较小尺寸的符号做为“小点”。下面是一个例子：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,20)
\multiput(10,2)(50,0){2}{\line(0,1){15}}
\put(10,14){\line(1,0){50}}\thicklines
\dottedline{3}(10,10)(60,10)
\dottedline[$\bullet$]{3}(10,5)(60,5)
\end{picture}
```



下面我们介绍在作图环境中使用的 join 类型的3个环境，它们是 dottedjoin 环境、dashjoin 环境和 drawjoin 环境。

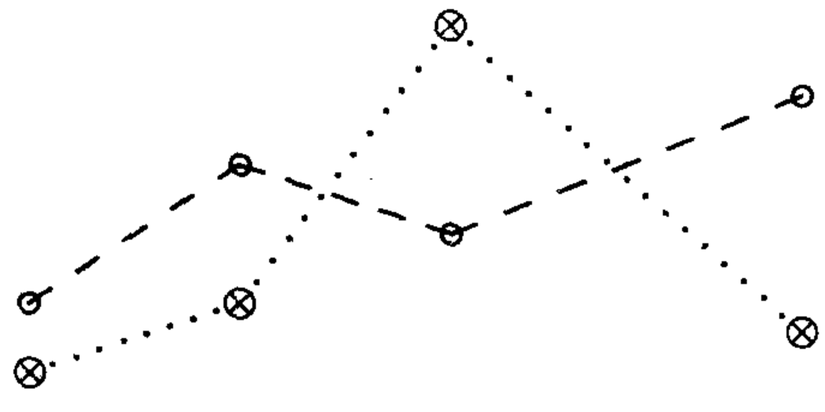
\begin{dottedjoin}[*dotcharacter*]{*dot-gap*} ··· \end{dottedjoin}
\begin{dashjoin}[*stretch*]{*dash-length*}[*dot-gap-in-dash*]
··· \end{dottedjoin}
\begin{drawjoin}[*stretch*] ··· \end{drawjoin}

上面这 3 个环境分别与前面介绍的 3 种线 \dottedline、\dashline 和 \drawline 相关。在每个环境中都是用命令

\jput(x,y){*object*}

将 *object* 居中放置在坐标 (x,y) 处，并且两个连续的 \jput 命令给出的坐标点会被与环境相对应的线条连接起来。*object* 通常要放在命令 \makebox(0,0){...} 构造的盒子中，除非它本身就是居中的（如 \circle 和 \circle*）。如果不放在这样的盒子中，则 *object* 的左下角参照点（而不是中心）将放在坐标点。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,40)
\begin{dottedjoin}{2} \thicklines
\jput(5,5){\plotot}\jput(20,10){\plotot}
\jput(35,30){\plotot}
\jput(60,8){\plotot}
\end{dottedjoin}
\begin{dashjoin}{2} \jput(5,10){\plotcc}
\jput(20,20){\plotcc}
\jput(35,15){\plotcc}
\jput(60,25){\plotcc}
\end{dashjoin} \end{picture}
```



其中用到的两条命令 \plotot 和 \plotcc 的定义是：

```
\newcommand{\plotot}{\makebox(0,0){$\otimes$}}
\newcommand{\plotcc}{\circle{1.5}}
```

命令 \jput 也可以在上面 3 个环境之外使用，不过这与 \put 没有什么区别。

6.4.2 epic 的扩展 —— eepic 宏包

eepic 宏包是对标准 LaTeX 作图环境以及 epic 宏包作图功能的扩展，放宽了它们在作图方面的某些限制。eepic 宏包允许用户画任意斜率的直线和任意大小尺寸的圆。不过在画有向线段时，斜率仍有同样的限制。也就是说，它也只能画斜率是 x/y 的有向线段，其中 x 和 y 是区间 $[-4,4]$ 中的整数。这个宏包中的画线命令

\line(x, y){length}

的用法与标准 LaTeX 中的命令 \line 的用法完全一样。但在这里 x 和 y 可以是任意能被 TeX 接受的整数。而且线段的长度也不再受标准 LaTeX 中线段长度最小值（大约是 3.5mm）的限制。

画圆和实心圆的命令

\circle{diameter} \circle*{diameter}

的用法也与标准 LaTeX 一样。但这里直径 *diameter* 可以是任意能被 TeX 接受的正数，并且所画的圆的直径恰好就是所给的 *diameter* 的值。

eepic 宏包还改写了标准 LaTeX 中画圆角矩形的命令

\oval(width,height){portion}

使得它的 4 个圆角的最大直径可以设置为任何值。最大直径由参数 \maxovaldiam 决定，它的默认值是 40pt。可以用 \renewcommand 命令重新设置它的值。

除了以上对标准 LaTeX 中的几条命令所做的改进之外，eepic 宏包还改进了 epic 宏包中的 \drawline、\dashline 和 \dottedline 命令以及所对应的三个 join 型环境，使得所画的图形效果更好，而且占用的内存更少。

另外，eepic 宏包还提供了一些用于作图的新命令：

\allinethickness{thickness}

这条命令设置其后所有线条的粗细程度，包括倾斜直线段、圆、椭圆、圆弧、圆角矩形和样条曲线。

\Thicklines

这条命令将其后的线条粗细程度设置得更粗些，大约是 \thicklines 的 1.5 倍。

$$\texttt{\textbackslash path}(x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$$

这是一条用直线段顺序连接各个坐标点的画线命令，但它要比 epic 宏包中的 \drawline 命令更快。这条命令主要用于画一些复杂的折线。

$$\texttt{\textbackslash spline}(x_1, y_1)(x_2, y_2)\cdots(x_n, y_n)$$

这是一条画样条曲线的命令，所画曲线是一条经过首尾坐标点的光滑曲线。中间的一些坐标都是控制点，就像 Bézier 曲线的中间坐标一样，但这里有更多的控制点。

\ellipse{*x-diameter*}{*y-diameter*}
\ellipse*{*x-diameter*}{*y-diameter*}

这是两条画椭圆的命令，第一条命令画的是空心椭圆，而带星号的命令画的是实心椭圆。其中 *x-diameter* 是水平方向上的直径参数，而 *y-diameter* 是垂直方向上的直径参数。当 *x-diameter* 和 *y-diameter* 的值相同时，这两条命令分别等同于画圆的命令 \circle 和 \circle*。

\arc{*diameter*}{*start-angle*}{*end-angle*}

这条命令画一条圆心为命令参照点，直径参数为 *diameter* 的圆弧，实际直径为 *diameter* 乘以单位长度。参数 *start-angle* 表示起点至圆心的线段与 x 轴的夹角，它的值必须在区间 $[0, \pi/2]$ 内。参数 *end-angle* 表示终点至圆心的线段与 x 轴的夹角，它的值必须介于 *start-angle* 和 *start-angle*$+2\pi$ 之间，因为大于这个值时，圆弧实际上是圆。这里角度的值用的是弧度单位，并且圆弧是朝着顺时针方向画的。

\filltype{*area-fill-type*}

这条命令用来指示如何填充实心圆和实心椭圆。参数 *area-fill-type* 的默认值是 black，即填黑色。另外也可以取 white（白色）和 shade（灰色）。

在使用 eepic 时，应注意如下几点：

1) 画直线段时，尽量避免使用 \line，最好使用 \drawline 命令，因为标准 LaTeX 的 \line 命令只支持有限几个斜率。

2) 如果所画的曲线较为复杂，应使用 \spline 命令，而不要用 \arc 命令。

3) 由于用命令 \dashline 画较长的虚线会占用较多的 TeX 内存，最好用 *stretch* 选项为负值的 \drawline 命令来画这种虚线。

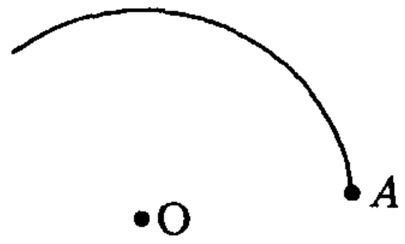
6.4.3 画任意曲线 —— curves 宏包

Ian Maclaine-cross 编写的 curves 宏包也是 LaTeX 作图环境功能的扩展，它使我们能画出各种复杂的光滑曲线，可以让所画的曲线经过预先指定的坐标点，曲线的粗细可以从 0.5pt 到 15pt（0.17mm 到 5.2mm）。也能画经过指定坐标点的光滑封闭曲线，以及直径较大的圆和圆弧，还可以在所画的曲线上添加各种符号。另外也能使图形朝某一方向伸展、压缩或旋转，因而也可以利用这个功能使圆变成椭圆。以下简要介绍其中的一部分命令。

\arc[*num*](x,y){*angle*}

这是一条画圆弧的命令，但是其用法与 eepic 宏包中的同名命令不同。这条命令以坐标为 (x, y) 的点为圆弧的起点，朝着逆时针方向画一个度数为 *angle* 的圆弧。圆弧所在圆的中心就是这条命令所在位置的参照点，一般由 \put 给出。整个圆弧是由一些更小的小圆弧段连接起来得到的，而选项 *num* 是一个自然数，表示用多少个点来组成这种小圆弧段，默认时所画的圆弧是实线的。另外，与 eepic 宏包中的 \arc 不同的是，这条命令中的角度用的是角度单位而不是弧度单位。角度参数 *angle* 的值是 120 就表示 120°。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,20)
\put(30,5){\arc(15,2){120}}
\put(45,7){\circle*{1}}
\put(46,6){\textit{A}}
\put(30,5){\circle*{1}}
\put(31,4){O}
\end{picture}
```



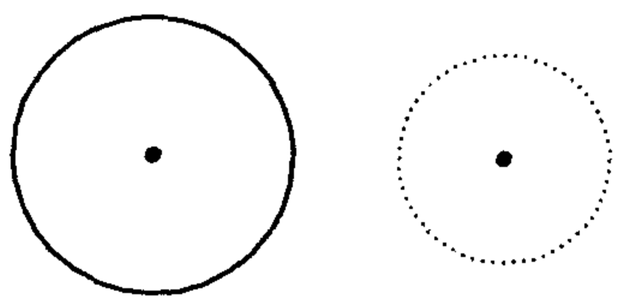
在上面这个例子中，因为我们已将画圆弧的命令 \arc(15,2){120} 放在 \put(30,5) 之中，所以圆弧的圆心在图形坐标系中的坐标应是 (30,5)，而圆弧的起点则在坐标为 (30,5)+(15,2)=(45,7) 的位置。

下面的命令

\bigcircle[*num*]{*diameter*}

类似于标准 LaTeX 的 \circle 画圆命令，它画一个直径为单位长度 \unitlength 乘以因子 *diameter* 的圆。选项 *num* 的意义同上。圆的中心就是命令所在的参照点，通常由 \put 给出。这条命令已突破了标准 LaTeX 画圆命令对直径的限制，它所画的圆的实际直径就是单位长度的 *diameter* 倍。另外要说明的是这里没有带星号形式的命令，因此不能用这条命令来画实心圆。

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,20)
\put(20,10){\bigcircle{20}}
\multiput(20,10)(25,0){2}{\circle*{1}}
\put(45,10){\bigcircle[4]{15}}
\end{picture}
```



上面这个例子中，画右边的圆时设置了选项 *num* 的值为 4，表示只用 4 个点来画小圆弧段，因此所画的圆是虚线圆。

curves 宏包定义了一个画经过所给点的封闭曲线的命令：

\closecurve[num]$(x_1, y_1, x_2, y_2, \cdots, x_n, y_n)$

这条命令画一条封闭曲线经过 n 个坐标点 (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n)。至少应给出 3 个坐标点。选项 num 的意义同上，仍表示用多少个点来组成画整条曲线的小圆弧段。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\closecurve(10,10,15,25,20,10)
\put(10,10){\circle*{1}}
\put(15,25){\circle*{1}}
\put(20,10){\circle*{1}}
\closecurve(35,10,35,20,50,20,60,10)
\end{picture}
```

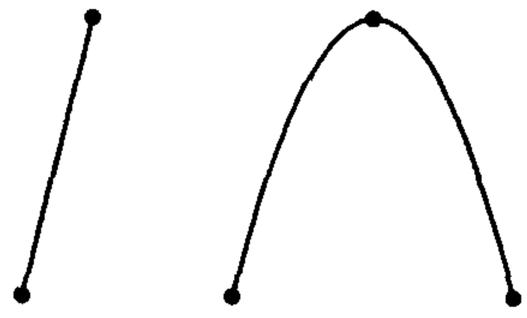
curves 宏包中还定义了命令 \curve、\closecurve 和 \tagcurve，它们都是用若干条抛物线线段按顺序连接所给的坐标点，并且两个相邻抛物线线段在连接处的切线是平行的。

下面的命令：

\curve[num]$(x_1, y_1, x_2, y_2, \cdots, x_n, y_n)$

表示画一条曲线经过 n 个坐标点 (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n)。至少应给出 2 个坐标点。如果只给出 2 个坐标点，那么这条命令就产生连接这 2 个坐标点的直线段。如果有 3 个坐标点，则产生一条抛物线顺序连接这 3 点。选项 num 也是自然数，表示除了坐标点之外曲线段上只画出 $num - 1$ 个点。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\put(15,5){\circle*{1}}
\put(20,25){\circle*{1}}
\curve(15,5,20,25)
\put(30,5){\circle*{1}}
\put(40,25){\circle*{1}}
\put(50,5){\circle*{1}}
\curve(30,5,40,25,50,5)
\end{picture}
```



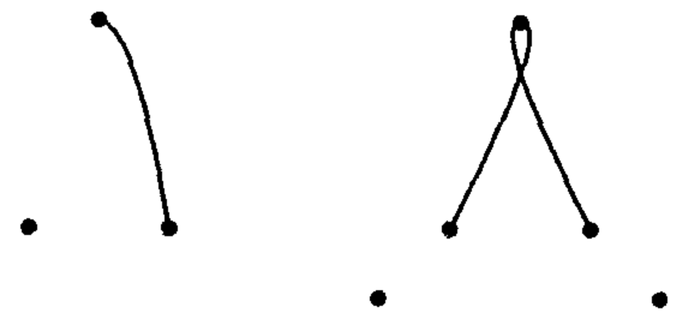
注意，当坐标点超过 3 个时，所画出的曲线可能不是光滑的，而在坐标点处出现尖点。

下面的命令：

\tagcurve[*num*]$(x_1,y_1,x_2,y_2,\cdots,x_n,y_n)$

表示用抛物线的线段连接 n 个坐标点 (x_1,y_1), (x_2,y_2), $\cdots$, (x_n,y_n)，使得在连接处尽量光滑，并且去掉最初和最后两段。如果只给出 3 个坐标点，则只画出最后一个抛物线线段。选项 *num* 的意义同上。例如：

```
\newcommand{\dian}{\circle*{1}}
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\tagcurve(10,10,15,25,20,10)
\put(10,10){\dian}
\put(15,25){\dian}
\put(20,10){\dian}
\tagcurve(35,5,40,10,45,25,50,10,55,5)
\put(35,5){\dian}\put(40,10){\dian}
\put(45,25){\dian}\put(50,10){\dian}
\put(55,5){\dian}
\end{picture}
```



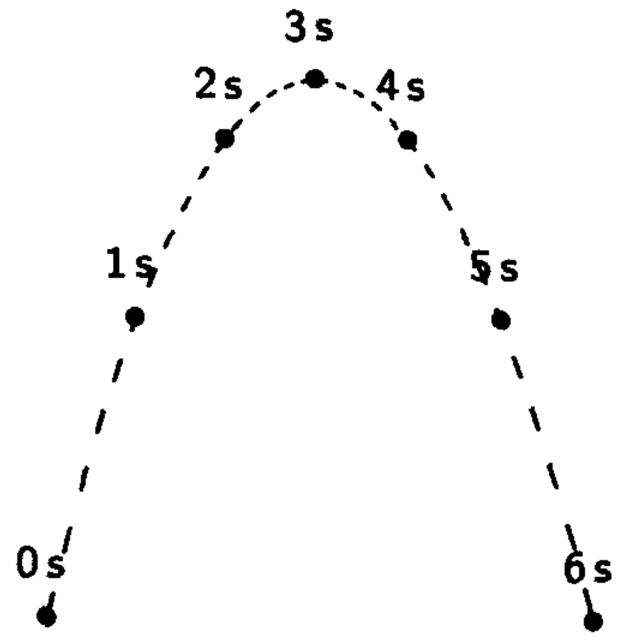
在这个例子中，为了方便起见我们定义了命令 \dian 来画较大的黑点。

curves 宏包还提供了下面的命令

\curvesymbol{*symbol*}

用它可以定义一种新符号作为画曲线的点。不过只有画曲线命令中的选项 *num* 为负整数时，才会用这条命令定义的符号来画曲线，而 *num* 绝对值就表示每个曲线段上所画符号的个数加 1（不含坐标点）。利用这条命令很容易在所画的曲线上添加一些符号。例如下面的图形展示了小球从抛出到落下的运动轨迹。

```
\setlength{\unitlength}{1mm}
\begin{picture}(50,45)
\curvedashes[1mm]{0,1,2}
\put(15,0){\curve(0,0, 19,39, 39,0)}
\curvedashes{} \newcounter{time}
\curvesymbol{\textsf{\thetime\,s}
             \addtocounter{time}{1}}
\put(15,4){\curve[-3](0,0, 19,39, 39,0)}
\curvesymbol{$\bullet$}
\put(15,0){\curve[-3](0,0, 19,39, 39,0)}
\end{picture}
```

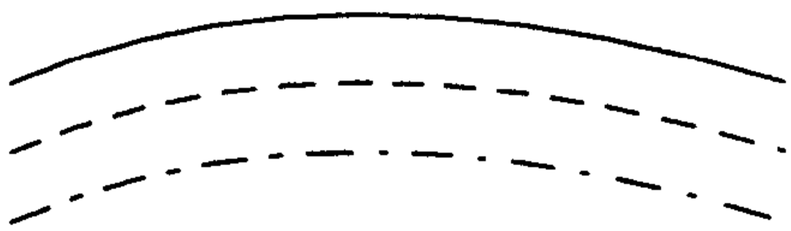


下面的 \curvedashes 命令：

\curvedashes[*unit-length*]{*blank-length*[,*dash-length*,$\cdots$]}

表示用一些较短的曲线段来画虚曲线，其中选项 *unit-length* 是长度（如 1mm, 1cm 等），代表单位长度，不能取负值，默认值由 \unitlength 给出。参数 *blank-length* 用来设置两个相邻小曲线段之间的空白。*dash-length* 设置小曲线段的长度。curves 宏包是按如下方法来画虚曲线的：假设花括号中的数字是 $a_1, a_2, \cdots, a_k$, 则系统从第一个坐标开始，画一段长为 a_1 的空白，再画一段长为 a_2 的小曲线段，接着画长为 a_3 的空白，再画一段长为 a_4 的小曲线段，这样交替地继续下去，当所给数字用完时，就又从头开始画长为 a_1 的空白，如此下去至坐标终点为止。注意，随着曲线弯度的增大，实际空白和小线段的长度都会按比例缩小。如果花括号中不给出任何数字，则表示按默认的方式画线，即画实线。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\curvedashes{0,3,2,0.5,2}
\curve(5,5, 30,10, 60,5)
\curvedashes{0,2,2}
\curve(5,10, 30,15, 60,10)
\curvedashes{}
\curve(5,15, 30,20, 60,15)
\end{picture}
```



另外，curves 宏包还提供了下面 4 个变换因子

\xscale, \yscale, \xscaley, \yscalex

用它们可以将由 curves 中命令构造的图形朝某一方向伸缩或者旋转。这 4 个变换因子的默认值分别是 1、1、0、0，但可以用命令 \renewcommand 重新设置它们的值。一旦重新设置了这些变换因子的值，其后的由 curves 中命令指明的坐标点 (x, y) 都将被新坐标 (x', y') 取代。新坐标和旧坐标之间的关系是：

$$x' = x \times \texttt{\textbackslash xscale} + y \times \texttt{\textbackslash xscaley},$$

$$y' = x \times \texttt{\textbackslash yscalex} + y \times \texttt{\textbackslash yscale}.$$

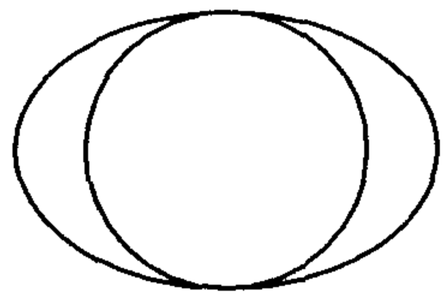
不难看出当只改变 \xscale 的值时，图形就朝着 x 方向伸缩；只改变 \yscale 的值时，图形就朝着 y 方向伸缩；当这 4 个变换因子构成一个旋转矩阵时，这种变换就是绕参照点的一个旋转。比如要把图形旋转一个角度 θ, 则可以重新设置 4 个变换因子的

值，使得

$$\begin{pmatrix} \backslash\text{xscale} & \backslash\text{xscaley} \\ \backslash\text{yscalex} & \backslash\text{yscale} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

当 θ 为正数时，向逆时针方向旋转，反之，当 θ 为负数时，则向顺时针方向旋转。一般情况下，这些因子所构成的变换是伸缩和旋转的复合。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,20)
\put(35,10){\bigcircle{20}}
\renewcommand{\xscale}{1.5}
\put(35,10){\bigcircle{20}}
\end{picture}
```

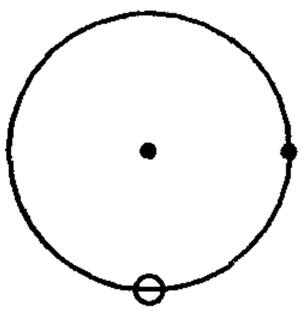


上面这个图形中椭圆是由圆向 x 轴方向拉伸 1.5 倍得到的。如果在改变了变换因子后使用命令

\scaleput(x,y){*object*}

则 *object* 实际上被放置在新坐标 (x', y') 处。例如：

```
\setlength{\unitlength}{1mm}
\begin{picture}(60,20)
\multiput(35,10)(10,0){2}{\circle*{1}}
\renewcommand{\xscale}{0}
\renewcommand{\xscaley}{1}
\renewcommand{\yscalex}{-1}
\renewcommand{\yscale}{0}
\put(35,10){\bigcircle{20}%
\scaleput(10,0){\bigcircle{2}}}
\end{picture}
```



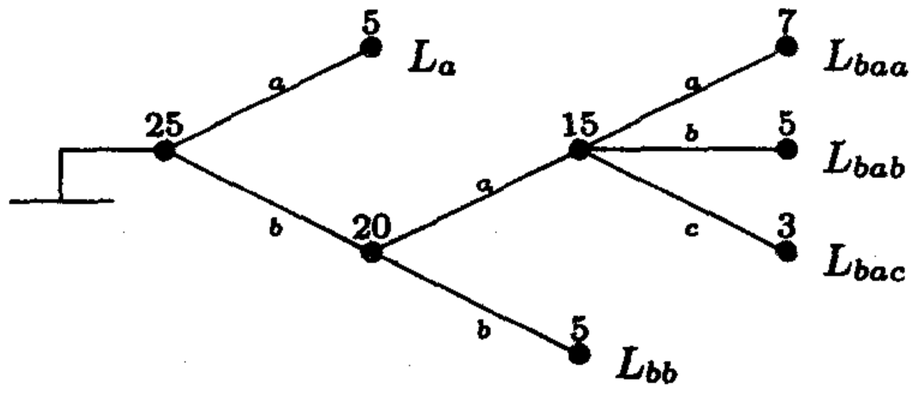
上面这个例子中重新定义了变换因子，使之成为一个朝顺时针方向旋转 90° 角的变换。由于将 \scaleput 放在 \put(35,10) 之中，所以 \scaleput 的旋转中心在 (35,10)，即左边的黑点处，本来应处于右边黑点（即坐标在 (35,10)+(10,0)=(45,10) 处的黑点）的小圆被旋转到当前的位置。

6.4.4 画树状分枝图 —— trees 宏包

trees 宏包是一个基于标准 LaTeX 作图环境的专门用来画树状分枝图表的宏包。它允许用户画有 2 个分枝或者 3 个分枝的树状图表，并且节点和枝条上都可以有标注。下

面就是一个使用 trees 宏包的例子：

```
\setlength{\unitlength}{3.5pt}
\begin{picture}(50,30)
\branchlabels{a}{b}{c}
\treeroot(10,15) 0.
\branch{2}{25}{0}:1,2.
 \leaf{5}{$L_a$} 1.
 \branch{2}{20}{2}:3,7.
   \tbranch{2}{15}{3}:4,5,6.
      \leaf{7}{$L_{baa}$}4.
      \leaf{5}{$L_{bab}$}5.
      \leaf{3}{$L_{bac}$}6.
 \leaf{5}{$L_{bb}$}7.
\end{picture}
```



树状分枝图表必须在作图环境中使用，其中定义了如下一些命令：

\branchlabels{*labela*}{*labelb*}{*labelc*}

这条命令设置枝条上的标注形式。*labela* 是第一个分枝的标注，*labelb* 和 *labelc* 分别是第二个和第三个分枝的标注。

\root$(z,y)$ *rootid.*

上面这条命令指出根的坐标 (x,y) 和它的标识符 *rootid.*。注意，其中圆括号、空格和句点都是必需的。

\branch{*steepness*}{*text*}{*branchid*}:*childa*,*childb*.

上面这条命令画含有 2 个子分枝的分枝节点。参数 *steepness* 的值可以取 0, 1, 2 或者 3，表示此节点的几个子分枝之间的角度的大小，值越小角度越大。参数 *text* 是写在此节点上方的标注。*branchid* 是此节点的标识符，它指明这个节点处于哪个分枝上。*childa* 和 *childb* 分别是此节点的两个子分枝的标识符。其中的冒号、逗号和最后的句点都是必要的。

\tbranch{*steepness*}{*text*}{*branchid*}:*childa*,*childb*,*childc*.

这条命令画含有 3 个子分枝的分枝节点。用法与 \branch 命令相同。

\leaf{*toptext*}{*sidetext*}{*leafid*}.

这是画枝叶的命令，其中 *leafid* 是此枝叶的标识符，*toptext* 和 *sidetext* 分别是写在枝叶上方和右边的标注。

画树状分枝图表时，一般都用 0 和正整数作标识符，当然若愿意的话也可以用其他符号作标识符。另外需要注意的是 trees 宏包中的 \root 命令与 amsmath 宏包中求根运算的同名命令相冲突。因此如果同时使用这两个宏包的话，最好将 trees 宏包或者 amsmath 宏包中的 \root 命令改为其他的名字。上面的例子中已经将 \root 重新定义成 \treeroot。

6.4.5 画条形统计图 —— bar 宏包

bar 是一个基于 LaTeX 作图环境的用来画条形统计图的宏包。它提供了一个在作图环境中使用的 barenv 环境，以及在这个新环境中使用的一些命令。利用这些命令可以画出各种 2 维和 3 维的条形统计图。下面逐条介绍其中的命令：

\bar{*height*}{*hatch-index*}[*description*]

这条命令画一个高为 \unitlength 乘以 *height* 的条形。*hatch-index* 是条形的序号，其值可以取 1 到 8 这 8 个数之一，每个数字代表一种条形，对应关系如下：

选项 *description* 可以用来对所画条形进行说明，其位置取决于命令 \setnumberpos 和 \setstyle.

\hlineon

这条命令将在条形区域的背景中画一些横线。

\legend{*hatch-index*{*legend-text*}

这条命令画出一个相应于 *hatch-index* 的小条形，并在其右边用 *legend-text* 表示这种条形所代表的意义。

\setdepth{*number*}

这条命令指示画 3 维的条形统计图，其中 *number* 是一个大于或等于 10 的数字，表示第 3 维的深度。

\sethspace{*fraction*}

这条命令定义两个相邻条形之间的空隙宽度，*fraction* 是一个对应条形宽度的因子，0.1 表示条形宽度的 0.1 倍。

`\setlinestyle{`*style*`}`

这条命令定义背景横线的形式，其值只能取 `solid` 或者 `dotted`，分别代表实线和虚线。

`\setnumberpos{`*position*`}`

这条命令设置描述条形的文字的位置，*position* 是位置参数，其值可取：

`empty`	无描述文字。
`axis`	描述文字在 x 轴的下方。
`down`	描述文字在相应条形的下方。
`inside`	描述文字在相应条形之中。
`outside`	描述文字在相应条形之外。
`up`	描述文字在相应条形的上方。

`\setprecision{`*digits*`}`

这条命令设置小数点后面的小数位数。

`\setstretch{`*factor*`}`

这条命令指示将条形向垂直方向拉伸多少倍。

`\setstyle{`*footstyle*`}`

这条命令设置其后的字体。

`\setwidth{`*number*`}`

这条命令定义条形的宽度为多少 pt。

`\setxaxis{`*origin*`}{`*end*`}{`*step*`}`

这条命令指示如何标注 x 轴上的坐标。第一个参数 *origin* 表示起点，它是第一个条形的 x 坐标；*end* 是终点，也就是最后一个条形的 x 坐标；*step* 是步长，即两个相邻条形的 x 坐标的差。需要注意的是，因为所画的条形的个数总是整数个，所以 *end* 减去 *origin* 必须是 *step* 的倍数。

`\setxname{`*x-label*`}`

这条命令定义的是对 x 轴上的坐标的说明，写在 x 轴右端的下方。

`\setxvaluetyp{`*type*`}`

这条命令设置用什么类型的量来标注 x 轴上的坐标。默认情况下是用数字来标注的。当 *type* 取 `month` 和 `day` 时，就分别用月份和日期来标注 x 轴上的坐标。不过，月份和日期都是德文的。如当 *type* 为 `month` 时，数字 1 和 2 分别对应 January 和 February；当 *type* 为 `day` 时，数字 1 和 2 分别对应 Monday 和 Tuesday。

`\setyaxis[`*offset*`]{`*origin*`}{`*end*`}{`*step*`}`

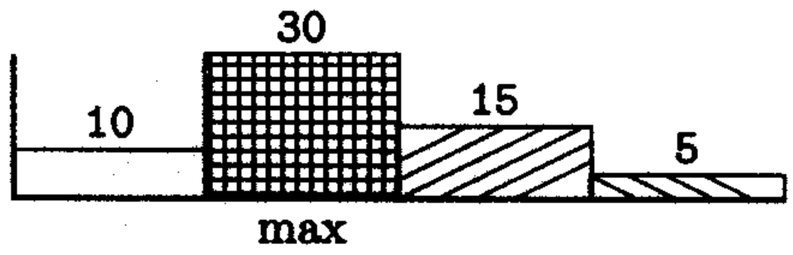
类似于 `\setxaxis`，上面这条命令用来设置标注 y 轴的坐标的方法。其中三个主要参数的意义与命令 `\setxaxis` 中的主要参数一样。另外，选项 *offset* 是加到 *origin* 和 *end* 上的偏差值，但它并不改变 y 轴上的各个坐标。

`\setyname{`*y-label*`}`

这条命令设置的是对 y 轴上坐标的说明，说明文字写在 y 轴顶端靠右的位置。

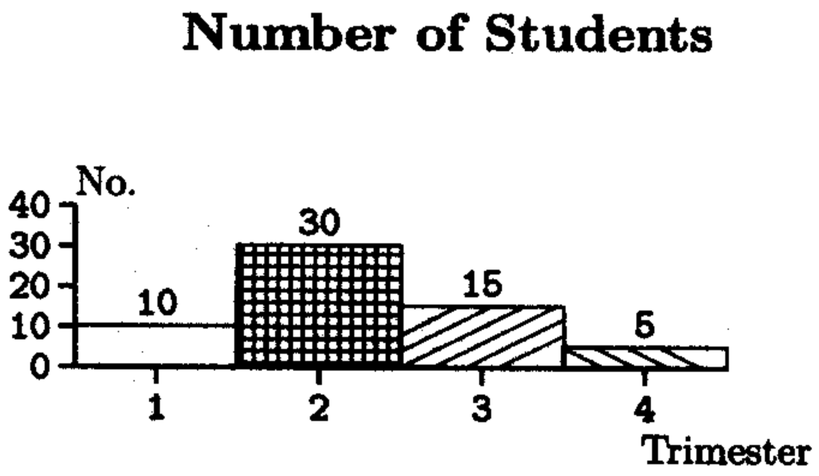
下面我们举几个例子来说明各种命令产生的效果。第一个例子是一个很简单的例子，它只使用了 `barenv` 环境的一些默认设置。

```
\begin{barenv}
\bar{10}{1}\bar{30}{4}[\texttt{max}]
\bar{15}{6} \bar{5}{7}
\end{barenv}
```



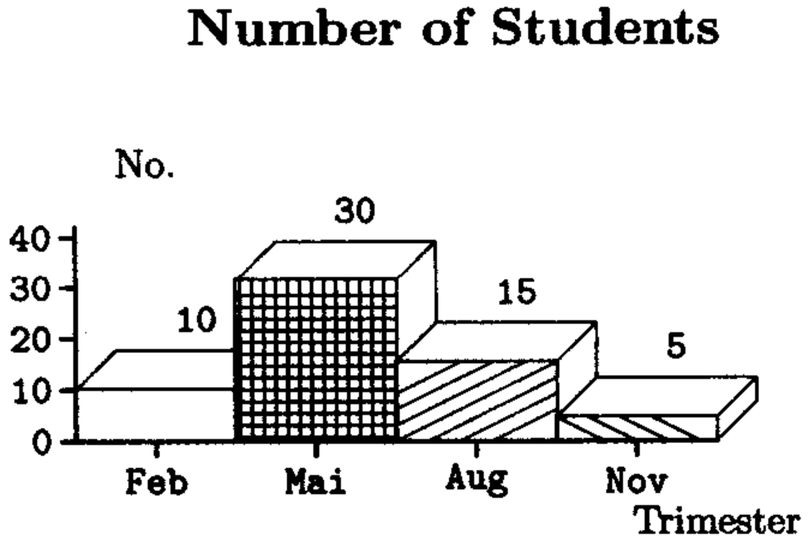
在下面的例子中，我们在统计图上加入了标题，并且在 x 轴上也对每个条形加上了坐标值。

```
\begin{center}
\textbf{Number of Students}\\[5mm]
\end{center} \begin{barenv}
\setxaxis{1}{4}{1}\setxname{Trimester}
\setyaxis{0}{40}{10}\setyname{No.}
\bar{10}{1}\bar{30}{4}\bar{15}{6}
\bar{5}{7} \end{barenv}
```



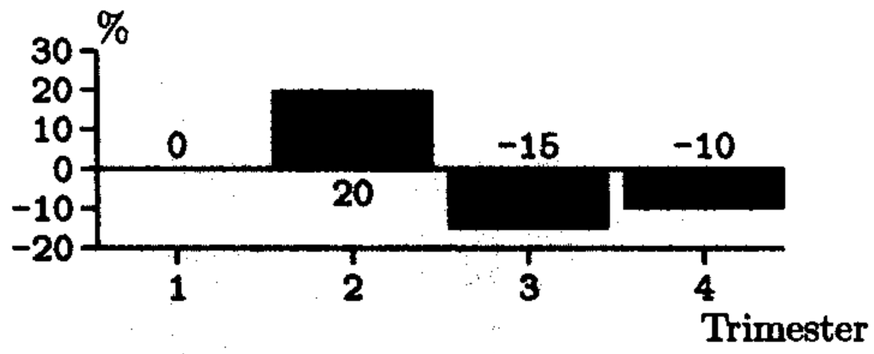
在下面的例子中，我们构造一个有 3 维效果的条形统计图，而且把条形拉长了一些使得看起来更明显了。另外，在 x 轴上用月份表示每一个条形的坐标。

```
\begin{center}
\textbf{Number of Students}\\[5mm]
\end{center}
\begin{barenv}
\setdepth{10}% 3维效果
\setstretch{1.4}% 拉伸条形
\setnumberpos{up}% 数字在条形上方
\setxvaluetyp{month} % x轴表示月份
\setxaxis{2}{12}{3} %起点为2步长为3
\setxname{Trimester}\setyaxis{0}{40}{10}
\setyname{No.}\bar{10}{1}\bar{30}{4}
\bar{15}{6}\bar{5}{7}
\end{barenv}
```



如果想强调各个条形所表示的数字的变化和差别，我们可以用如下形式的统计图：

```
\begin{center}
\textbf{Student Number Variation}
\end{center}
\begin{barenv}
\setxaxis{1}{4}{1}\setxname{Trimester}
\setyaxis{-20}{30}{10}\setyname{\%}
\sethspace{0.1} \setnumberpos{axis}
\bar{0}{1}\bar{20}{8}
\bar{-15}{8}\bar{-10}{8}
\end{barenv}
```



下面是一个比较复杂但接近实际的例子，输出结果见图 6.6。

```
\begin{center}
\textbf{\Large Share price for Company \textsf{XyZ}}\\[5mm]
\begin{barenv}
\setstretch{1.6}
\setnumberpos{down}
\setwidth{25}  \setdepth{10}
\setstyle{\small\bfseries}
\setxname{Year} \setyname{US Dollars}
\setstyle{\small\itshape}
\setxaxis{1982}{1992}{1}
\setstyle{\small\bfseries}
\setyaxis[10]{0}{160}{10}
\setlinestyle{dotted} \hlineon \setnumberpos{axis}
\bar{70}{5}  \bar{105}{4} \bar{100}{4} \bar{150}{6}
\bar{125}{6} \bar{140}{6} \bar{115}{4} \bar{105}{4}
```

```
\bar{105}{4} \bar{90}{5}  \bar{70}{5}
\end{barenv}
\end{center}
\par\vspace{2\baselineskip}
 Yield: \legend{6}{Good} \qquad \legend{4}{Moderate}\qquad \legend{5}{Bad}
```

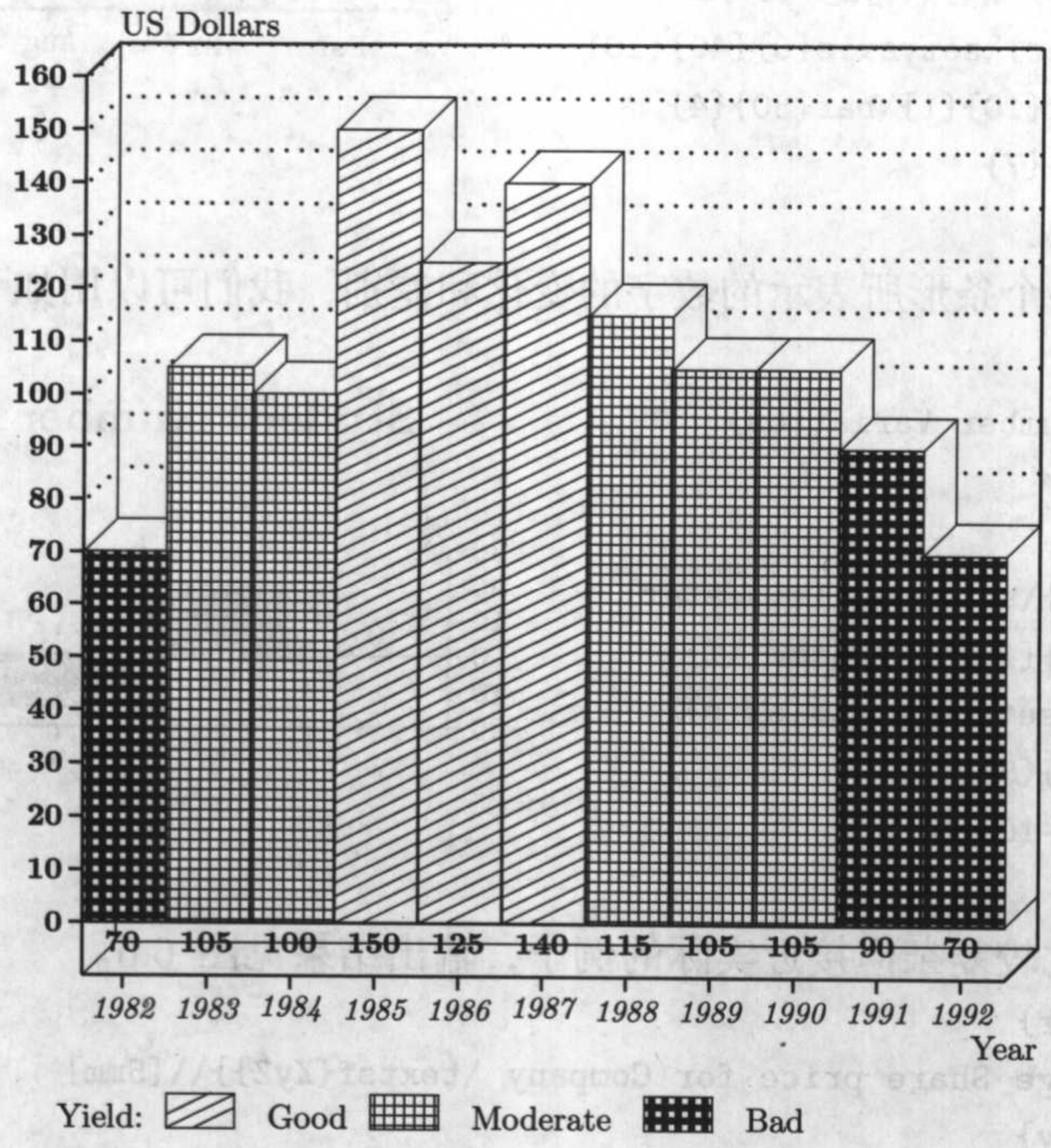


图 6.6 bar 宏包的例子

6.4.6 其他作图宏包

在电子工程方面，Adrian Johnstone 编写了 lcircuit 宏包，其中定义了一套在 LaTeX 作图环境中使用的逻辑集成电路符号。这些符号中有 4 个方向的逻辑端口、供电针、发射端口、电容、电阻等大多数电路符号，如图 6.7 所示。这些符号图形都是用前面提到的 bézier 曲线构造的。

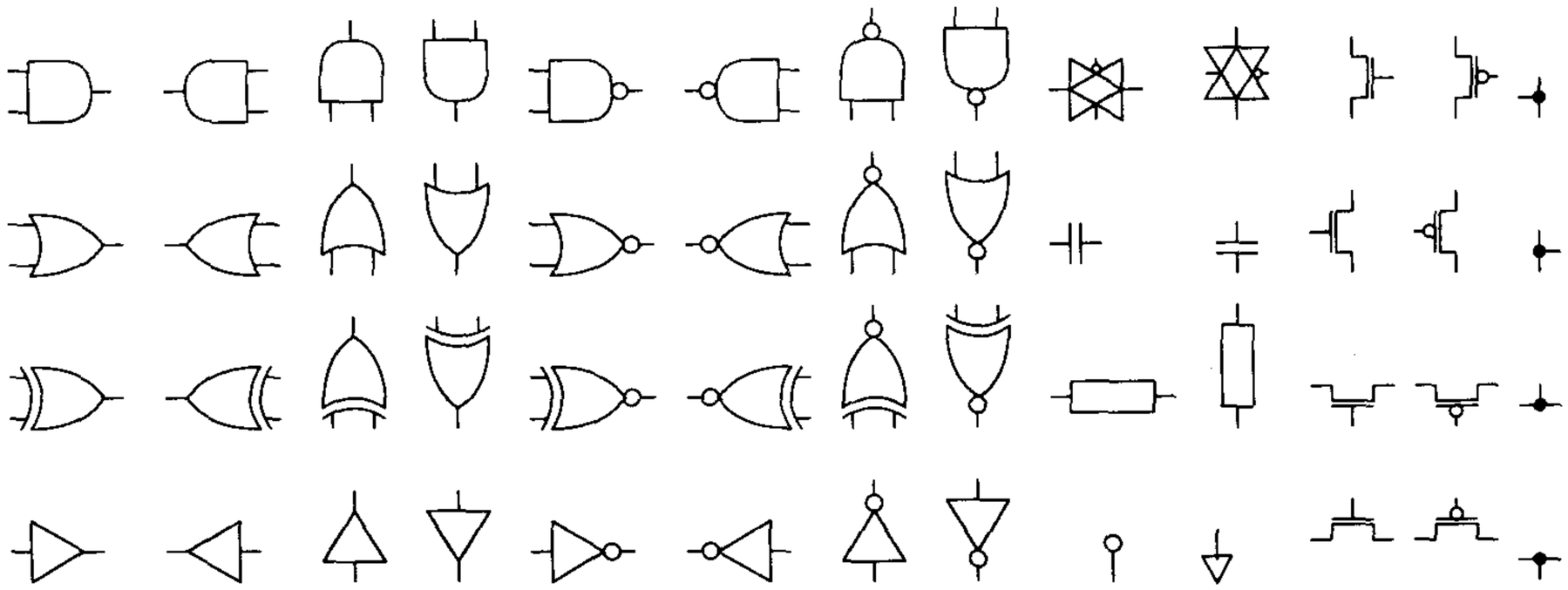


图 6.7 picture 环境中使用的电路符号

在理论物理文献中经常使用 Michael Levine 编写的 Feynman 宏包来画一种称为 Feynman 图表的图形。这个宏包允许在 LaTeX 作图环境中使用一些高端命令语句来画 Feynman 图表中的各种质点线和节点群。图 6.8 就是用这个宏包画出来的一个典型 Feynman 图表。

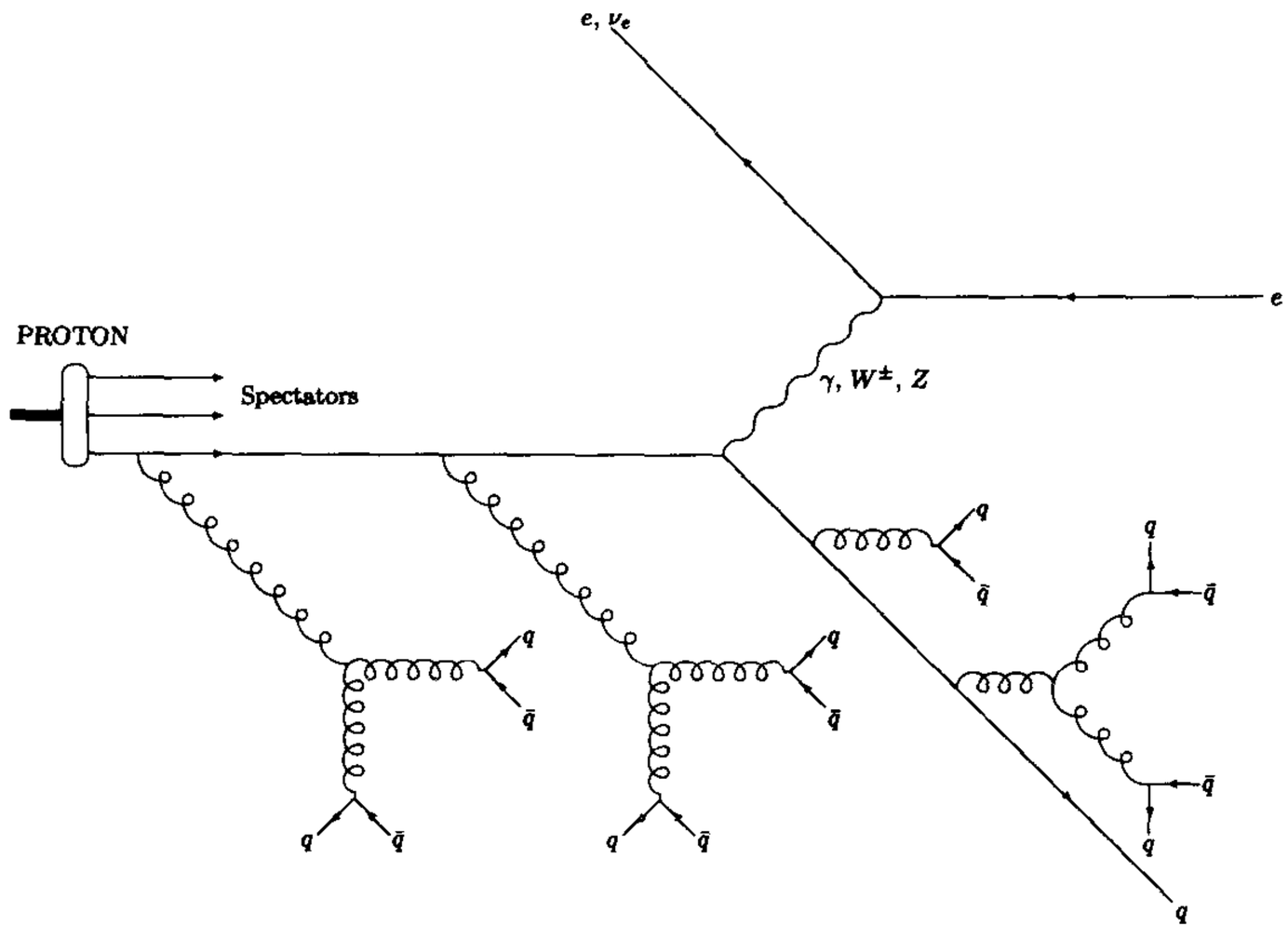


图 6.8 Feynman 宏包的例子

在有机化学方面，Roswitha Haas 和 Kevin O'Kane 开发了一个 ChemTeX 宏包，它利用 LaTeX 的作图环境定义了许多新命令来画各种分子符号。另外，Mats Dahlgren 编写了一个 chemsym 宏包，其中对化学周期表中的每一个元素以及氘、甲基、乙醛等都定义了对应的命令，使得在文稿中很容易输入这些符号。

第7章 插 图

一本好的书籍通常都是图文并茂的。特别是科技文献经常需要插入图片来说明事例。本章将阐述如何在 LaTeX 文稿中插入图形以及如何对所插入的图形进行旋转或缩放等必要的处理。

7.1 图 形 格 式

由于当 Knuth 编写 TeX 系统时，还没有 PS/EPS、JPEG、GIF 等现今流行的一些图形格式，因此 dvi 文件并不直接支持这些格式。不过，TeX 允许 dvi 文件中包含 `\special` 命令来向 dvi 处理程序传递信息，这就使得 TeX 和 LaTeX 能够经由这种方式使用大多数的图形格式。因为 dvi 文件经常被转换为 EPS 文件，而且 EPS 也是最早被引入 LaTeX 的图形格式，所以 LaTeX 对它的支持也是最好的。

7.1.1 EPS 图形

EPS 格式的图形是用 Encapsulated PostScript 语言来描述的。Encapsulated PostScript 语言实际上是 PostScript(PS) 语言的一个子集。这种语言能够描述图形和文本。PS 文件和 EPS 文件的主要区别在于：EPS 文件仅仅使用了部分特定的 PostScript 命令，另外，EPS 文件必须含有一个 BoundingBox 行来确定图形的大小。通常 PostScript 文件的第一行标明该文件的类型，接下来的几行被作为 header 或 preamble 的注释行（PostScript 的注释符也是%）。注释行中有一行就定义了 BoundingBox。BoundingBox 中有 4 个整数值，按顺序分别代表图形左下角的 x 坐标，左下角的 y 坐标，右上角的 x 坐标，右上角的 y 坐标。例如，下面的代码

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: gnuplot
%%DocumentFonts: Times-Roman
%%BoundingBox: 50 50 410 302
%%EndComments
```

是一个由 gnuplot 生成的 EPS 文件的前 5 行。这个 EPS 图形的左下角的坐标是 (50,50)，右上角的坐标是 (410,302)。这里坐标单位是 PostScript point，等于 1/72 英寸，在 TeX 和 LaTeX 中用 bp 表示这种单位。因此，上面这个例子表示的图形的自然宽度为 5 英寸，自然高度为 3.5 英寸。有时在插入图形时需要计算该图形的 BoundingBox 值。一个比较方便的方法是用 ghostview 或者 gsview 将图形打开，当鼠标在图形上移动时就

会显示鼠标位置（以页面左下角为参照点）的坐标。记下图形左下角和右上角的坐标就可以确定图形的 BoundingBox 值。

7.1.2 格式转换工具

因为 LaTeX 对 EPS 格式的图形支持最好，所以常常需要将其他格式的图形转换为 EPS 格式的图形。下面列出了一些免费软件和共享软件可以用于将各种图形格式转换为 EPS 格式：

• ImageMagick 是一个很好的免费图形转换工具，支持大多数操作系统。它可以将许多非 EPS 格式的图形转换为 EPS 格式的图形，并且可以对图形进行旋转、清晰化、重定尺寸、改变颜色等操作，还可以进行其他一些特殊处理。此软件可以在 `http://www.imagemagick.org/` 下载。

• xv 是一个运行在 X-Windows 环境下的共享软件，可以用来观看和转换图形。它也可以处理许多图形格式，但是不支持命令行方式，因此不能用它来实现图形格式的即时转换功能。有关 xv 的信息可参见下面的网页：

`http://www.sun.com/sunsoft/catlink/xv/note.html`

• wmf2eps 是一个运行在 Windows9.x/NT 下的免费软件。它可以将 WMF 格式的图形转换为 EPS 格式的图形。可以在 `ftp://www.ctan.org/tex-archive/support/` 取得该软件。

• ImageCommander 是一个 Windows3.1/95/NT 下的共享软件。利用它可以将 GIF、JPEG、PICT、WMF 等多种格式的图形转换为 EPS 和其他格式的图形。

• jpeg2ps 是一个免费的 C 语言程序，它可以将 JPEG 格式的图形转换为 Level 2 EPS 图形。Level 2 EPS 图形支持压缩的二进制图形，因而能够制作出比传统的 EPS 图形质量更好并且文件尺寸更小的图形。jpeg2ps 也可以在上面的 ftp 网址获得。

• bmeps 是一个免费的命令行方式的图形转换软件，它可以将 PNG, JPEG, TIFF 格式的图形转换为 Level 2 EPS 图形，能与 LaTeX 很好地结合起来，并且可以在多种操作系统中安装。目前 MiKTeX 系统已包含了该软件。最新的版本可在上面的 ftp 网址下载。

7.2 插 图 命 令

为了方便用户在 LaTeX 文件中插入图形，LaTeX 3 小组开发出一个比早期的一些图形宏包更有效更稳定的图形宏包套件：LaTeX graphics boundle。它包括标准的 graphics 宏包和扩展的 graphicx 宏包。这两个宏包都有命令 `\includegraphics`，不过版本不同。graphicx 版的 `\includegraphics` 命令采用命名机制，使用起来比较方

便，而且支持图形的缩放和旋转；而在 graphics 版中则要借助于命令 \scalebox 和 \rotatebox 才能达到同样的效果。要使用 graphicx 宏包，首先需在源文件的导言部分输入

```
\usepackage{graphicx}
```

这条命令。下面这条命令

```
\includegraphics[option]{filename}
```

将名为 *filename* 的图形插入到命令出现处，其中 *option* 是选项，两个选项之间必须用逗号隔开。表 7.1 中列出了可以使用的选项及其相应的意义。因为 \includegraphics 命令不会结束当前段落，所以能够将图形置于文本段落之中如 和  。如果插入的图形的文件名没有扩展名，那么命令 \includegraphics 会根据命令

```
\DeclareGraphicsExtensions
```

所指明的扩展名列表自动为它加上一个扩展名。另外，默认的扩展名列表不包含空的扩展名，因此，\includegraphics{file} 不会插入名为 file 的图形，除非空的扩展名已被添加到扩展名列表中。例如，下面的命令

```
\includegraphics[width=3in,angle=45]{file.eps}
```

将名为 file.eps 的图形按比例缩放到宽度为 3in 并逆时针旋转 45° 角再插入到文稿中。如果使用 \textwidth 这类参数来指定宽度而非用 3in 这样的固定值，将会使 LaTeX 文件更具通用性。

表 7.1 命令 \includegraphics 的选项

一般选项	说　　明
height	图形的高度（可取任何 TeX 度量单位）。
totalheigh	图形的总体高度（可取任何 TeX 度量单位）。
width	图形的宽度（可取任何 TeX 度量单位）。
scale	图形的缩放因子（可取正十进制数）表示图形自然大小的倍数，如 scale=2 表示所插入的图形的大小为图形自然大小的 2 倍。
angle	图形旋转的角度（可取正或负十进制数）。以度为单位，正数表示逆时针方向。
bb	设定图形的 BoundingBox 值。如 bb=10 20 100 200 设定图形的左下角在 (10,20), 右上角在 (100,200)。因为 \includegraphics 会自动从 EPS 文件中获取 BoundingBox 值，所以一般不使用这个选项，但当 EPS 文件中的 BoundingBox 值丢失或出错时还是有用的。
origin	指定图形绕哪一点旋转，默认时表示绕图形的参考点旋转。

续表

剪切选项	说 明
viewpoint	指定图形可以被显示的部分，如同 BoundingBox 的值一样也是由 4 个数字组成，前两个数字表示显示部分的左下角坐标，后两个数字表示显示部分的右上角坐标，但这里的坐标是相对于 BoundingBox 的左下角的。例如，如果 BoundingBox 的值是 50 50 410 302, 那么 viewpoint=50 50 122 122 将显示以图形的左下角为左下角的 1in 大小的区域， viewpoint=338 230 410 302 则显示以图形的右上角为右上角的 1in 大小的区域。只有使用了 clip 选项时，才能使显示区域以外的部分不被显示出来。
trim	这个选项也可以用来指定图形被显示的部分。所给出的 4 个数字分别表示从图形的左边缘、下边缘、右边缘、上边缘被截去的值，正数表示从边缘截去的大小，而负数则表示从边缘加上的大小。例如，trim=1 2 3 4 表示被显示的区域在图形内部且距左边缘 1bp, 距下边缘 2bp, 距右边缘 3bp, 距上边缘 4bp。只有使用了 clip 选项时，才能使显示区域以外的部分不被显示出来。

布尔选项	说 明
noclip	这是默认选项，表示显示整个图形，即使有些部分在显示区域之外。
clip	当使用此选项时，将不显示 viewpoint 和 trim 中指定的视图之外的部分。
draft	表示草稿方式。使用此选项时，将只显示图形的 BoundingBox 和文件名，这使得显示和打印的速度加快。如果在源文件的导言部分使用 draft 宏包选项 `\usepackage[fraft]{graphicx}`，会使整个文件中图形都被以草稿的方式插入。
final	通常用它来覆盖 `\usepackage[fraft]{graphicx}`，这是默认选项，使得图形被显示出来。
keepaspectratio	没有设定这个选项时，若给定图形的高度（总体高度）和宽度可能会使图形以不对称缩放来满足所给的高度和宽度。若设定了这个选项，图形在缩放时会保持原有的宽高比例，并尽可能使图形满足所设定的宽和高，但不会超过任一个给定的值。

7.3 图形的旋转和缩放

除了上一节中所介绍的插图命令 `\includegraphics` 之外，graphicx 宏包中还包含了另外 4 条用于旋转和缩放任意 LaTeX 对象（如图形和文本）的命令，它们是用于缩放对象的命令 `\scalebox`、`\resizebox` 和 `\resizebox*` 以及用于旋转对象的命令 `\rotatebox`。因为 `\includegraphics` 命令本身已经带有缩放和旋转对象的选项，所以这 4 条命令实际上很少使用。例如下面 3 条插图命令

```
\scalebox{2}{\includegraphics{file.eps}}
\resizebox{4in}{!}{\includegraphics{file.eps}}
\rotatebox{45}{\includegraphics{file.eps}}
```

可以分别用

```
\includegraphics[scale=2]{file.eps}
\includegraphics[width=4in]{file.eps}
\includegraphics[angle=45]{file.eps}
```

替代。尽管这两种方法结果相同，但在实际使用中最好还是用后一种方法，因为这种方法在生成 PostScript 文件时有更高的效率和更快的速度。

7.3.1 scalebox 命令

下面这条命令

```
\scalebox{水平缩放因子}[垂直缩放因子]{对象}
```

将对象进行缩放，使得缩放后的对象的宽度等于对象原始宽度与水平缩放因子的乘积，而高度等于原始高度与垂直缩放因子的乘积。如果不给出垂直缩放因子，那么就按照所给的水平缩放因子在保持宽高比的条件下进行缩放。如果缩放因子为负数，则除了将对象缩放之外还会将对象反射。例如：

```
\begin{minipage}{\linewidth}
\centering
\scalebox{1.5}{放大的文字} \\
\scalebox{.7}{缩小的文字}\\
\scalebox{1}[-1]{纵向反射的文字}\\
\scalebox{-1.5}{放大且双向反射的文字}
\end{minipage}
```

放大的文字
缩小的文字
纵向反射的文字
放大且双向反射的文字

7.3.2 resizebox 命令

下面两条命令

```
\resizebox{宽度}{高度}{对象}
\resizebox*{宽度}{总体高度}{对象}
```

将对象按照所给的宽度、高度或者总体高度进行缩放。当参数宽度或者高度中的一个被设置成感叹号！时，对象将按照另一个参数进行缩放，并保持宽高的比例不变。不带星号的 \resizebox 与带星号的 \resizebox* 之间的区别仅在于前者的第二个参数表示的

是对象的高度，而后者的第二个参数表示的是对象的总体高度。在使用高度选项时要特别小心，尽管它的值常常与总体高度的值相等。比如当对象的深度为零时，高度的值实际上也就是总体高度的值，此时使用高度选项一般不会有什么问题。但当对象的深度不为零时，使用高度选项而不是总体高度选项就有可能导致图形显示的大小不正确。当插入一个 EPS 格式的图形时，特别是要对此图形进行旋转和缩放时，区分高度和总体高度是很重要的。下面这条命令

```
\resizebox{4in}{!}{\includegraphics{file.eps}}
```

将一幅名为 file.eps 的图形的宽度缩放成 4in, 且宽高的比例不变，然后插入到文稿中。标准 LaTeX2e 中参数 \width、\height、\totalheight、\depth 可用来表示对象的原始尺寸。因此，命令

```
\resizebox{2in}{\height}{对象}
```

将对象的宽度缩放成 2in，但保持原来的高度不变。

7.3.3 rotatebox 命令

下面的这条命令

```
\rotatebox[选项]{角度}{对象}
```

将对象旋转一个给定的角度。角度的值为正数表示按逆时针方向旋转，而负数表示按顺时针方向旋转。选项 可以用来设定对象围绕哪一点（旋转中心）旋转，默认时表示对象围绕其参照点进行旋转。可以用下面的两种方法给出旋转中心：

1) 给出相对于对象参照点的坐标，比如选项 [x=2cm,y=1cm] 表示旋转中心相对于参照点的坐标为 (2cm,1cm)。

2) 用 [origin=位置]来表示旋转中心，其中位置是由表示 3 个水平位置的字母 l、c、r（分别代表左、中、右）之一与表示 4 个垂直位置的字母 t、c、B、b（分别代表顶部、中部、基线、底部）之一的组合。于是，有 12 种可供选择的位置。如图 7.1 所示。

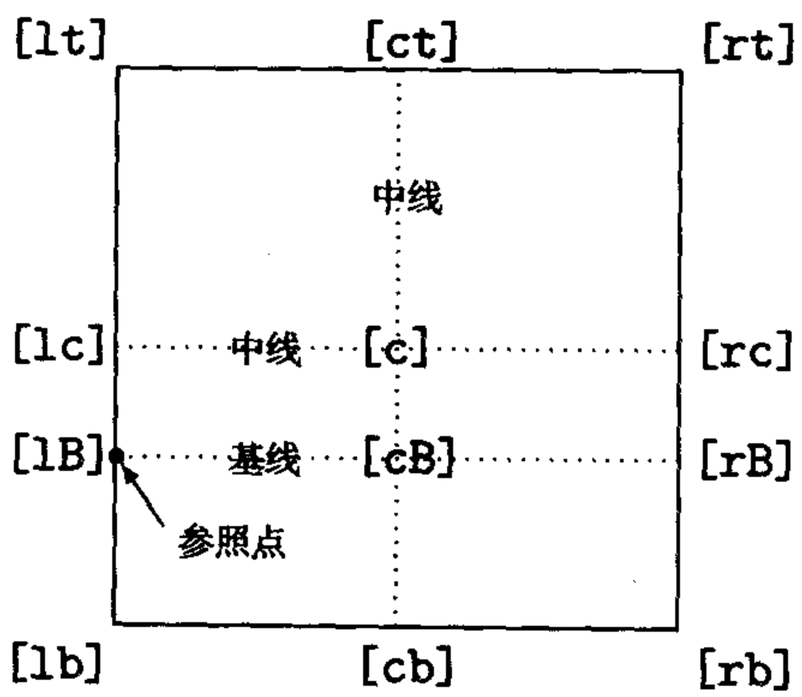


图 7.1 旋转中心的位置

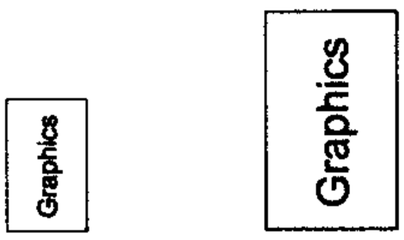
例如：选项[lt] 表示左上角；[lb] 表示左下角；[rt] 表示右上角；[rb] 表示右

下角；[cB] 表示基线的中点；[lB] 表示参照点的位置；这里需要说明的是：当两个字母组合时字母的顺序并不重要，[br] 与 [rb] 表示同一点。字母 c 表示的是水平位置的中点还是垂直位置的中点，要看它与哪个字母结合。如果只给出一个字母，那么另一个字母将被假设为 c。因此 [c] 与 [cc] 是相同的选项，而 [l] 与 [lc] 相同，[r] 与 [rc] 相同等。

7.3.4 选项的顺序

当使用 \includegraphics 命令将图形插入文稿时，改变所给选项的顺序有可能导致不同的输出结果。事实上，LaTeX 总是从左至右依次处理 \includegraphics 的选项的。例如：

```
\begin{center}
\includegraphics[angle=90,%
  totalheight=1cm]{graphics.eps}
  \hspace{1cm}
\includegraphics[totalheight=1cm,%
  angle=90]{graphics.eps}
\end{center}
```



其中第一条命令先将图形逆时针旋转 90°，再缩放图形使其总高度为 1cm；而第二条命令先缩放图形使其总高度为 1cm，再将图形逆时针旋转 90°。所得结果明显不同。

7.3.5 旋转对大小的影响

在插图命令中所给出的宽度和高度并不表示图形的实际大小，而只是表示图形的 BoundingBox 的大小。当使用插图命令将图形进行旋转和缩放时，理解这一点是很重要的。例如，利用下面三条插图命令

```
\begin{center}
    \includegraphics[totalheight=1in]{rose.eps}
    \includegraphics[angle=45,totalheight=1in]{rose.eps}
    \includegraphics[angle=90,totalheight=1in]{rose.eps}
\end{center}
```

将一个名为 rose.eps 的 EPS 图形插入进来，就得到

中间那个被逆时针旋转 45° 角的图形明显变小了。这似乎有点奇怪，但只要看一看它们的 BoundingBox 也就明白了。在上面的三个插图命令中都添加一个选项 `draft` 就得到

从这里就可以看出这三个图形的 BoundingBox 的总体高度其实是一样的，都被缩放到了 1in。

7.3.6 旋转图形的对齐

在 TeX 中每一个图形都被当作一个盒子，它有自己的高度、深度、宽度以及参照点。当图形未被旋转时，图形的深度为零，因而其高度也就是它的总体高度，其参照点就是位于盒子左下角的点。但是当图形旋转时这些参数就可能发生变化，图 7.2 显示了图形经过旋转后的高度、深度、宽度以及参照点的变化情况。

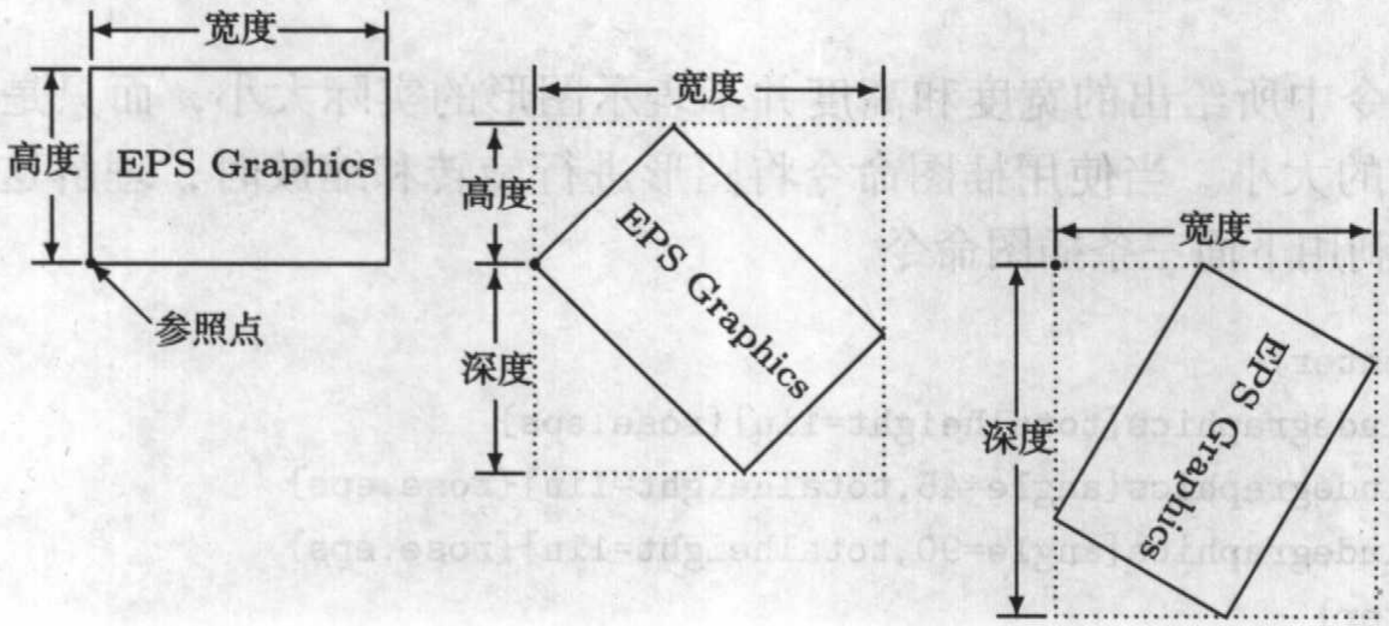
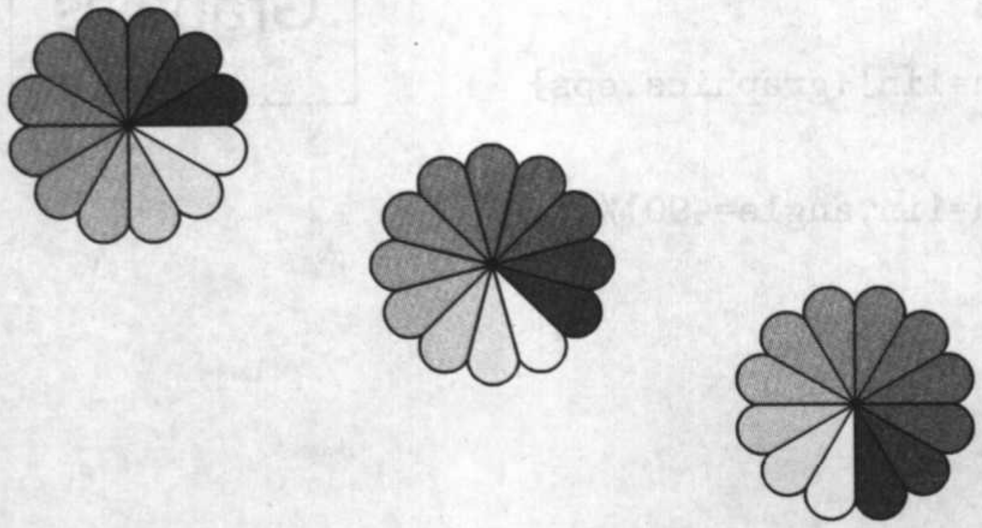


图 7.2 图形旋转示例

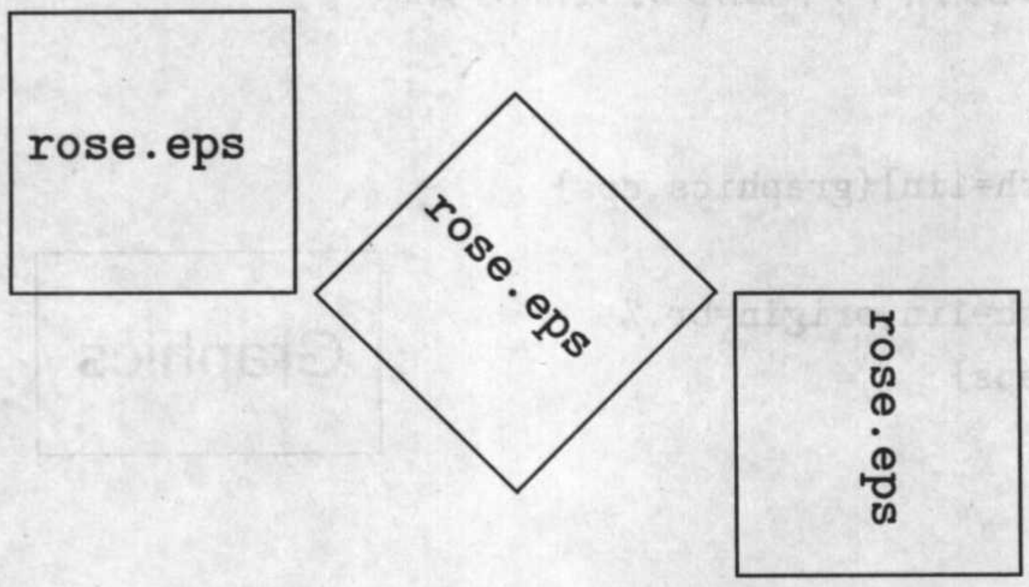
上面中间的图形是由左边的图形顺时针旋转 45° 得到的，其高度减小了但深度和宽度都增加了。右边的图形是由左边的图形顺时针旋转 120° 得到的，此时高度已经减小到零。正是由于这种情况，当几个被旋转的图形放在一起时可能不能很好地对齐。例如下面命令

```
\begin{center}
    \includegraphics[totalheight=.8in]{rose.eps}
    \includegraphics[totalheight=.8in,angle=-45]{rose.eps}
    \includegraphics[totalheight=.8in,angle=-90]{rose.eps}
\end{center}
```

的输出结果为



虽然看起来 3 个图形没有对齐，但通过它们的 BoundingBox:



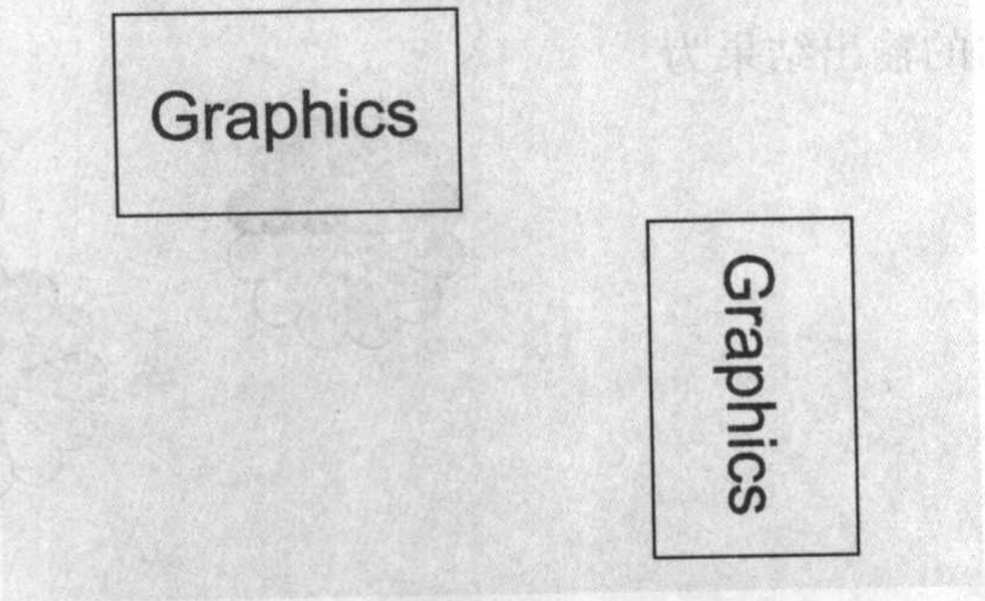
可以看到它们的参照点还是在同一条水平线上的，也就是说这 3 个图形是按照参照点对齐的。若要将图形按照它们的中心对齐，可以利用 origin 选项。例如：

```
\begin{center}
    \includegraphics[totalheight=.8in]{rose.eps}
    \includegraphics[totalheight=.8in,origin=c,angle=-45]{rose.eps}
    \includegraphics[totalheight=.8in,origin=c,angle=-90]{rose.eps}
\end{center}
```

的输出结果为

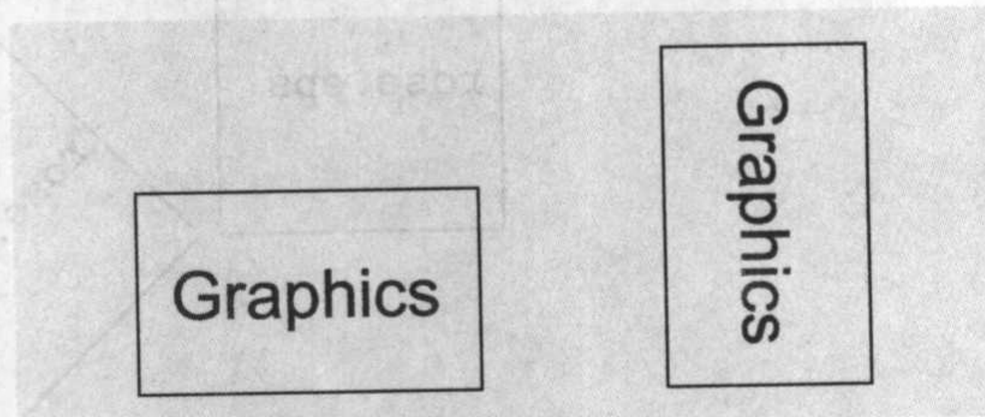
下面的例子中第 2 个图形是由第 1 个绕其参照点顺时针旋转了 90° 得到的，基线因而移到其顶部。于是第 2 个图形的顶部便与第 1 个图形的底部对齐。

```
\begin{center}
\includegraphics[width=1in]{graphics.eps}
\hspace{1cm}
\includegraphics[width=1in,angle=-90]%
{graphics.eps}
\end{center}
```



在下例中第 2 个图形是由第 1 个图形绕其右下角顺时针旋转了 90° 得到的，此时基线仍在其底部，因而此例中两个图形的底部对齐。

```
\begin{center}
\includegraphics[width=1in]{graphics.eps}
\hspace{1cm}
\includegraphics[width=1in,origin=br,%
angle=-90]{graphics.eps}
\end{center}
```



7.3.7 小页中的图形

在使用中常常需要将图形置于小页环境中，这也是很有效的方法。当若干个小页并列在一起时， LaTeX 会使它们的参照点对齐。默认的情况下，小页的参照点是它的左边线的中点。小页环境有一个选项用来改变其参照点的位置，选项值可取字母 b、c、t，其中 c 是默认值。它们的意义如下：

b 它使小页的参照点与小页中最后一行的参照点对齐。

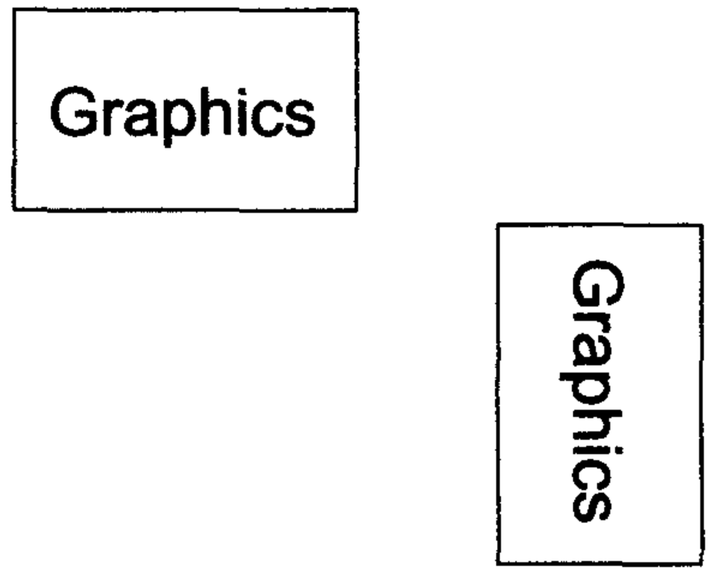
c 它使小页的参照点是小页左边线的中点。

t 它使小页的参照点与小页中第一行的参照点对齐。

要注意只有当小页的最后一行恰好在小页底部时，b 才表示小页的参照点与小页中最后一行的参照点对齐。同样，只有当小页的第 1 行恰好在小页顶部时，t 才表示小页的参照点与小页中第 1 行的参照点对齐。

当小页中只有 1 行时，b 和 t 的效果是一样的。例如：

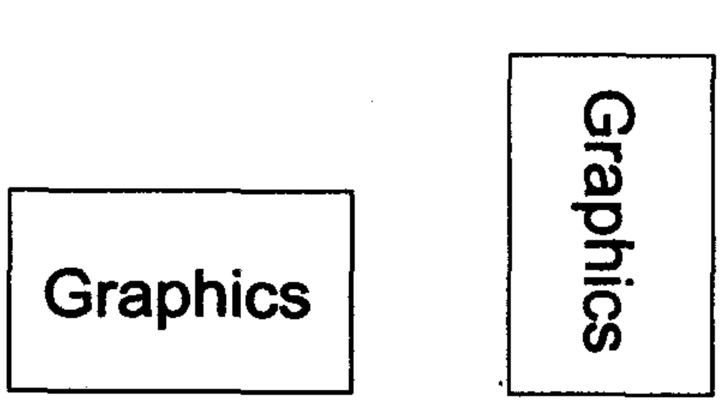
```
\begin{center}
  \begin{minipage}[b]{.45\linewidth}
  \centering \includegraphics%
  [width=1in]{graphics.eps}
  \end{minipage}%
  \begin{minipage}[b]{.45\linewidth}
\centering\includegraphics[width=1in,%
  angle=-90]{graphics.eps}
  \end{minipage}
\end{center}
```



如果将上面例子中的选项 b 改为 t 所输出的结果是一样的。

当我们想把两个小页的底部对齐时，只需使小页的底部成为它们的基线即可。如果在小页中的图形后面插入一个高度和深度都是零的不可见线条，那么小页的 b 选项就会使底部成为小页的基线。这种零高度零深度的不可见线条可以用命令 \par\vspace{0pt} 得到。因为这条零高度零深度的不可见线条的基线就是小页的底部，所以小页的 b 选项就会使小页的底部对齐。例如：

```
\begin{center}
\begin{minipage}[b]{.45\linewidth}
\centering \includegraphics[width=1in]%
{graphics.eps}\par\vspace{0pt}
\end{minipage}
\begin{minipage}[b]{.45\linewidth}
\centering\includegraphics[width=1in,%
angle=-90]{graphics.eps}\par\vspace{0pt}
\end{minipage}
\end{center}
```

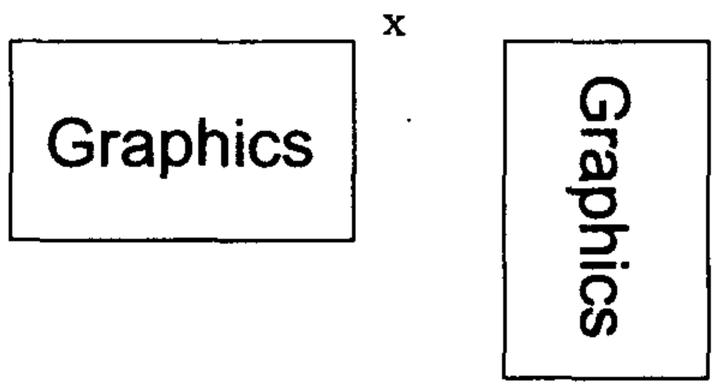


此时两个图形的底部已经对齐，选项 b 确实起了作用。

类似地，若想把两个小页中的图形按顶部对齐，必须在小页中的最上面加入一条零高度零深度的不可见线条。这样，小页的 t 选项就会使小页的基线在小页的顶部。可以在小页中的最上面插入命令 \vspace{0pt} 得到一条零高度零深度的不可见线条。因为

这条不可见线条的基线就是小页的顶部，所以小页的 t 选项就会使小页中的图形按顶部对齐了。例如：

```
\begin{center}
\begin{minipage}[t]{.40\linewidth}
\centering \vspace{0pt}
\includegraphics[width=1in]%
{graphics.eps}\end{minipage} x
\begin{minipage}[t]{.40\linewidth}
\centering\vspace{0pt}
\includegraphics[width=1in,angle=-90]%
{graphics.eps}
\end{minipage}
\end{center}
```



注意此时小页顶部实际上是与当前文本行的基线对齐的，若要让小页的顶部与当前文本行的顶部对齐就需要用一个适当的负长度值来代替 \vspace{0pt} 中的 0pt。

7.3.8 小页或盒子的背景图案

\boxput 是 fancybox 宏包中提供的一条命令，用它可以将文字或图案放置在小页或盒子的背景中。这条命令的用法是：

\boxput(x,y){背景内容}{前端内容}
\boxput*(x,y){前端内容}{背景内容}

上面两条命令将前端内容放置在背景内容的上面，其中 (x,y) 是可选项，它决定将前端内容的中心放置在背景内容的哪个位置。这个坐标表示以背景内容中心为原点的坐标，不过对 x 来说，1 个单位是背景内容 宽度的一半；对 y 来说，1 个单位是背景内容高度的一半。x 和 y 可取任何数字。

(0,0) 表示将前端内容的中心放置在背景内容的中心。这是默认设置。
(0,1) 表示将前端内容的中心放置在背景内容上边缘的中点。
(0,-1) 表示将前端内容的中心放置在背景内容下边缘的中点。
(1,0) 表示将前端内容的中心放置在背景内容右边缘的中点。
(1,1) 表示将前端内容的中心放置在背景内容的右上角。
(1,-1) 表示将前端内容的中心放置在背景内容的右下角。
(-1,0) 表示将前端内容的中心放置在背景内容左边缘的中点。
(-1,1) 表示将前端内容的中心放置在背景内容的左上角。
(-1,-1) 表示将前端内容的中心放置在背景内容的左下角。

背景内容和前端内容可以是文字、图形、小页等。在下面的例子中，我们将一个包含两行前端文字的段落盒子放置在以较淡文字为背景的图形上：

```
\definecolor{mygray}{gray}{0.75}
\boxput(0,0){\textcolor{mygray}{%
\zihao{0} 背景图形}}{\parbox{4cm}%
{\centering\large 前端文字\\
 前端文字}}
```

前端文字
前端文字

其中命令 \definecolor 和命令 \textcolor 分别用来定义颜色和选择字体颜色（见 7.5.2 节）。\zihao 是自定义命令（见 9.3.6 节），用来选择中文字体的字号。

7.4 压缩图形及非 EPS 图形

本质上不能将一个压缩的图形或者一个非 EPS 格式的图形直接插入到 LaTeX 文件中。不过当使用 dvips 程序时，用户可以定义一条命令使得在插入图形之前对图形进行必要的操作。这个操作如果是解压缩的命令，就可以使用压缩的图形文件了；若这个操作是一个图形格式转换命令，那么就可以使用非 EPS 格式的图形了。在此之前必须告诉 LaTeX 所要插入的图形是什么格式的以及如何对它进行操作。

7.4.1 扩展符及操作方式声明

下面的命令

```
\DeclareGraphicsExtensions{扩展符列表}
```

指定了在没有提供图形文件的扩展名的情况下，LaTeX 可以自动为它加上一个扩展名的扩展名列表。LaTeX 总是按顺序读入所给出的扩展名。例如源文件中有如下命令时

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,{}}
```

命令 \includegarphics{file} 将先查找名为 file.eps 的文件，其次查找 file.ps，再其次查找 file.eps.gz，最后查找不含扩展名的 file。

下面的命令

```
\DeclareGraphicsRule{ext}{type}{sizefile}{command}
```

指定 \includegarphics 如何按照文件的扩展名对图形文件进行操作，源文件中可以允许多次使用这条命令。这条命令要求操作系统具有管道功能，如 Unix、Linux 等（DOS 没有管道功能）。命令中

ext 表示文件的扩展名。

type 表示扩展名所对应的图形格式。

sizefile 表示包含图形 BoundingBox 的文件的扩展名。如果这个选项为空，那么必须在 \includegraphics 命令中用 bb 选项指明图形的 BoundingBox 值。

command 表示作用于图形文件的操作命令，此选项常常空着。命令前面必须有一个朝后的单引号“‘”（不是常用的朝前的单引号“’”）。到目前为止，只有 dvips 能处理这样的命令。

例如，命令

```
\DeclareGraphicsRule{.eps.gz}{eps}{.bb}{‘gnuzip -c -c #1}
```

指定任何以 .eps.gz 为扩展名的文件都是压缩的 EPS 文件，该文件的 BoundingBox 信息存放在以 .bb 为扩展名的文件中，并且用命令 gunzip -c 来解压缩。由于 LaTeX 不能从压缩的文件中读取 BoundingBox 信息，以 .bb 为扩展名的文件必须是没有压缩的纯文本文件。

\DeclareGraphicsRule 命令还允许使用 * 来表示任何未知扩展名，如：

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

表示任何未知扩展名的文件都被认为是 EPS 文件。

7.4.2 使用压缩的 EPS 文件

在 LaTeX 中使用压缩的 EPS 文件，通常有如下几个步骤：

1) 创建一个所要插入的图形的 EPS 文件（如 file.eps）。有许多方法可用来创建或将图形转换为 EPS 文件。2) 查看创建的 EPS 文件，获取其 BoundingBox 信息并将它保存在一个纯文本文件中（如名为 file.eps.bb 的文件）。3) 压缩 EPS 文件，比如在 Unix 中可用命令：

```
gzip -9 file.eps
```

得到压缩的文件 file.eps.gz。这里的 -9（或者 -best）选项表示最佳压缩。一旦得到压缩的 EPS 文件后就可以删除原先的 EPS 文件。4) 在源文件的导言部分使用 \usepackage{graphicx} 调用图形宏包，并在正文部分插图处使用 \includegraphics 命令插入压缩的 EPS 文件。在此之前还要用 \DeclareGraphicsRule 命令，使得 LaTeX 知道如何处理特殊扩展名的文件。

下面是一个例子：

```
\documentclass[dvips]{article}
```

```
\usepackage{graphicx}
\begin{document}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{`gunzip -c #1}
\begin{figure}
\centering  \includegraphics[width=3in]{file.eps.gz}
\caption{压缩的 EPS 图形} \label{fig:compressed:eps}
\end{figure}
\end{document}
```

在本例子中，\DeclareGraphicsRule 命令可以不用，这是因为系统在 dvips.def 中已经定义过了。但是如果不用 gunzip 程序来解压缩而用其他程序，那么这条命令不能省略。

7.4.3 使用非 EPS 图形文件

虽然 EPS 格式的图形能够很容易被插入到 LaTeX 文件中，但是对于非 EPS 格式的图形文件并不是简单地将文件名的扩展符替换一下就行的。事实上，必须借助图形格式转换工具才能将非 EPS 格式的图形文件插入到 LaTeX 文件中，而且还要求操作系统有管道功能。尽管如此，由于一般说来 EPS 文件的尺寸比较大，而某些其他格式（如 JPEG 格式）的图形尺寸比较小，我们有时还是希望避免直接对 EPS 文件进行操作。通常在 LaTeX 文件中插入非 EPS 格式的图形文件，需要如下几个步骤（以插入 JPEG 格式图形文件 file.jpg 为例）：

1) 找一个支持命令行方式的 JPEG 到 EPS 的图形转换程序，如 jpeg2ps。

2) 获取图形文件的 BoundingBox 值，并将其存放到名为 file.bb 的非压缩的纯文本文件中。可以用 ebb 程序（它是 dvipdfm 中一个用来计算非 EPS 格式图形文件的 BoundingBox 值的程序，许多 TeX 系统中都有）直接得到 file.bb 文件，也可以先将 file.jpg 转换为 file.eps, 通过查看这个 EPS 文件而得到 BoundingBox 值，然后再删除这个 EPS 文件。

3) 在 LaTeX 源文件中，用下面的命令

```
\DeclareGraphicsRule{.jpg}{eps}{.bb}{`jpeg2ps #1}
```

告诉 LaTeX 系统如何处理 JPEG 格式的图形文件。并且在正文部分使用插图命令 \includegraphics{file.jpg} 将图形插入到文件中。此时，LaTeX 会从 file.bb 读入 BoundingBox 值并通知 dvips 用 jpeg2ps 将 file.jpg 转换为 EPS 文件。

7.5 颜色的使用

LaTeX 图形宏包套件中还包含一个 color 宏包，在导言部分调用这个宏包，我们就可以在排版 LaTeX 文件时使用颜色了。

7.5.1 色样体系

LaTeX 对颜色的支持是建立在色样体系之上的。不同的驱动程序所支持的色样体系也可能不同，但大部分驱动程序都支持以下几种色样体系：

rgb 三原色（red 红，green 绿，blue 蓝）。使用 3 个 0 到 1 之间的十进数分别表示红、绿、蓝的深浅，两个数字之间用逗号隔开。

cmyk 四分色（cyan 青，magenta 品红，yellow 黄，black 黑）。使用 4 个 0 到 1 之间的十进数分别表示青、品红、黄、黑的深浅，两个数字之间也要用逗号隔开。

gray 灰度。使用 1 个 0 到 1 之间的十进数表示灰色的深浅。

named 使用名称调用颜色，如：“JungleGreen”。并非所有驱动程序都支持这种方式。所用的颜色名称必须被驱动程序识别或者已事先由某些命令定义好了（见 `color.dtx` 中的说明）。

7.5.2 定义和使用颜色

black(黑)、white(白)、red(红)、green(绿)、blue(蓝)、cyan(青)、magenta(品红)、yellow(黄)，这些颜色在 color 宏包已经定义过了。如果要用其他没有定义的颜色，必须先用下面的命令

```
\definecolor{名称}{色系}{颜色定义}
```

来定义。例如，

```
\definecolor{light-blue}{rgb}{0.8,0.85,1}
\definecolor{mygrey}{gray}{0.75}
```

定义了名称分别为 light-blue 和 mygrey 的两种颜色。

使用颜色有两种方法。一种是使用已有名称的颜色，另一种是直接使用颜色。下面的命令

```
\color{name}
```

将它后面的文字的颜色改为名称 *name* 所代表的颜色，直到当前群组结尾。其中 *name* 必须是已经定义过的。这是一个声明形式的命令，如同 `\bfseries` 一样。

也可以使用下面类似于 \textbf 的参数形式命令：

```
\textcolor{name}{一些文字}
```

这条命令将一些文字的颜色改为由名称 *name* 所代表的颜色，同样 *name* 必须是已经定义过的。这条命令等价于 {\color{*name*}一些文字}。

下面两条命令

```
\color[色系]{颜色定义}
\textcolor[色系]{颜色定义}{一些文字}
```

的作用与前面两条一样，只不过它们不是用已有名称的颜色，而是直接使用所给出的颜色。比如，\textcolor[rgb]{1,0.2,0.3}{一些文字} 将一些文字的颜色改为由三原色中 {1,0.2,0.3} 所对应的颜色。

7.5.3 页面及盒子的背景色

如果要将整个页面的背景都设置成某种颜色而不是白色，可以用命令

```
\pagecolor{name}
```

这条命令将当前及后续页面的背景颜色设置成颜色名称 *name* 所表示的颜色。因为这是一条整体命令，所以如果想从某页开始还原成白色背景需使用命令\pagecolor{white}。

如果要将页面中某个盒子的背景设置成别的颜色可以用下面两条命令

```
\colorbox{name}{一些文字}
\fcolorbox{name1}{name2}{一些文字}
```

其中第一条命令输出一个包含一些文字的盒子，这个盒子的背景颜色是颜色名称 *name* 所表示的颜色，如 **淡蓝色背景的盒子** 。第二条命令输出一个包含一些文字的盒子，这个盒子的背景颜色是颜色名称 *name2* 所表示的颜色，并且有一个 *name1* 颜色的边框。如 **淡蓝色背景的盒子** 。这两条命令与 \fbox 一样也使用参数 \fboxrule 和 \fboxsep 来定义边框的粗细和盒子中文字到边缘的距离。结合命令 \colorbox 和 \textcolor 还可以定义一个排版反白文字的命令：

```
\newcommand{\fanbai}[1]{\setlength{\fboxsep}{0.2pt}%
    \colorbox{black}{\textcolor{white}{\bfseries #1}}}
```

\fanbai{例} 反白字的例子。　　**例** 反白字的例子。

7.6 多次插入同一图形

当一幅 EPS 图形在 LaTeX 文件中被多次使用，它的 EPS 代码就会多次出现在最后生成的 PS 文件中。比如将公司的标志图案插入到每页的页眉、页脚或边空中，再比如要在每页都插入一幅同样的水印图案等。如果处理不好就可能使生成的 PS 文件很大，也可能使打印机不能正常打印。这一节介绍一种改进的方法用来在 LaTeX 文件中多次插入同一图形。

通常有 4 种方法用来在 LaTeX 文件中多次插入同一图形。

1) 在图形出现处每次都使用命令 `\includegraphics{file.eps}`。这种方法会使 LaTeX 每次都去找寻和读入 `file.eps` 文件，并且 EPS 图形命令会在最后生成的 PS 文件反复出现，从而使 PS 文件变得很大。

2) 将图形储存在 LaTeX 盒子中，当要插入图形时就调用这个盒子。即在源文件正文部分开始时使用

```
\newsavebox{\mygraphics}
\sbox{\mygraphics}{\includegraphics{file.eps}}
```

然后在需要插图的地方使用命令 `\usebox{\mygraphics}` 调用图形。还可将这条命令置于 `\scalebox` 或者 `\rotatebox` 中将图形缩放或旋转。这种方法能节约 LaTeX 处理图形的时间，因为 LaTeX 只需一次性找寻和读入图形文件，不过不会使最后生成的 PS 文件变小。

3) 当 EPS 文件是一个矢量图形文件时，就可以写一个 PostScript 命令来画这个图形。从而在需要插入图形的地方调用这个 PostScript 命令就可以了。7.6.1 节中介绍了写 PostScript 命令的过程。用这种方法生成的 PS 文件只包含一次 PostScript 命令，因此文件会比较小。不过在打印这个文件时，一遇到这个图形打印机就读入相应的 PostScript 命令，而且这些命令都储存在打印机内存中，从而有可能使打印机内存耗尽，而不能打印文件。另外，使用这种方法时， LaTeX 仍需多次找寻和读入包含 PostScript 命令的文件，因此并不能节约 LaTeX 处理图形的时间。

4) 与第三种方法一样定义一个 PostScript 命令来画这个图形，并且将这条命令储存在 LaTeX 盒子中。在插图处调用这个盒子。这种方法既能使最后生成的 PS 文件较小，又能节约 LaTeX 处理图形的时间。

7.6.1 定义 PostScript 命令

这一节介绍如何从一个矢量图形的 EPS 文件来定义画这幅图的 PostScript 命令。这种方法不适合包含位图的 EPS 文件。

要将一个矢量图形的 EPS 文件转换为 PostScript 命令，必须将这个 EPS 文件分成两个文件。其中一个文件包含 PostScript 字典和图形命令，另一个文件包含 EPS 文件头（header）信息以及已定义的 PostScript 命令。虽然不同程序生成的 EPS 文件的文件头是不同，但大体格式都差不多。例如，一个由 xfig 程序创建的 EPS 文件有如下的形式：

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep 3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog
$F2psBegin
...
$F2psEnd
```

这里 ··· 表示没有列出的命令。一般来说，一个 EPS 文件包含 3 个部分：

1) 由 % 引导的 header 命令。它包含文件头信息。

2) Prolog 部分，它开始于 `/$F2psDict 200 dict def`，并由 `%%EndProlog` 结尾。这个部分定义了在 EPS 文件中用于 PostScript 字典的命令。在上面这个例子中字典的名字是 `$F2psDict`。当然也可以用其他的名字。

3) 最后一部分是用于画图的命令。

假设上面这个 EPS 文件以 `file.eps` 命名，我们需要建立两个文件：`file.h` 和 `file.ps`。其中 `file.h` 包含：

```
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog
/MyFigure {
$F2psBegin
...
$F2psEnd
} def
```

而 file.ps 包含：

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep 3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
$F2psDict begin MyFigure end
```

file.h 中定义了字典和 PostScript 命令 /MyFigure，而 file.ps 包含 header 信息，并调用在 file.h 中定义的命令。特别重要的一点是 file.ps 必须包含 %!PS ··· 这一行以及含有 BoundingBox 的行。然后在 LaTeX 文件中按照类似于

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}
\begin{document}
...
\includegraphics[width=2in]{file.ps}
...
\includegraphics[totalheight=1in]{file.ps}
...
\end{document}
```

的方法多次插入同一个图形。注意这里没有使用原来的文件 file.eps，因此可以删除它。由于在 file.h 中定义的命令只被加载了一次，最后生成的 PS 文件会很小。然而每次插入图形时，LaTeX 仍会找寻和读入文件 file.ps。

下面先将文件 file.ps 储存在 LaTeX 盒子中，然后在插图时调用这个盒子：

```
\documentclass{article}
\usepackage{graphicx}
\special{header=file.h}
\newsavebox{\mygraphic}
\sbox{\mygraphic}{%
\includegraphics[width=2in]{file.ps}}
\begin{document}
...
\usebox{\mygraphic}
...
```

```
\resizebox*{1in}{!}{\usebox{\mygraphic}}
...
\end{document}
```

与前面一样仍输出一个 2in 宽的图形和一个有 1in 总体高度的图形。这次不仅能生成比较小的 PS 文件，LaTeX 也只需一次找寻和读入 `file.ps` 这个文件。

7.6.2 页眉或页脚中的图形

在页眉或页脚中插入图形的一个比较容易的方法是使用 3.5.4 节中介绍的 fancyhdr 宏包。例如，有一个名为 `file.eps` 的 EPS 图形需要插入到页眉中，可以按照如下的方法来做。首先用 7.6.1 节中的方法将 `file.eps` 分为两个文件：`file.h` 和 `file.ps`。然后用类似于如下的命令：

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}
\renewcommand{\headheight}{0.6in}% 页眉应足够高
\renewcommand{\textheight}{7.5in}
\special{header=file.h}% 定义 PostScript 图形命令
\newsavebox{\mygraphic}
\sbox{\mygraphic}{\includegraphics[totalheight=0.5in]{file.ps}}
\pagestyle{fancy}
\fancyhead{} % 清除不必要的页眉
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{} % 清除不必要的页脚
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\begin{document}
...
\end{document}
```

将图形放置在每一使用 `fancy` 版式的页面页眉的左边，同时页眉下方有一条粗 0.5pt 的横线，页码在页脚的中间。注意必须将页眉的高度设置的足够大使得能容纳所插入的图形。对于双页排版的文件，如果想让图形插入到左页的左边和右页的右边，那么可以使用 `\fancyhead` 命令的 `E`、`O` 选项。比如可以将上面例子中的命令

```
\fancyhead[L]{\usebox{\mygraphic}}
```

替换为

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

还要注意到上面的设置并没有改变 plain 版式的页面，因此标题页和每章的开始都还是原来的样子，不会出现图形。如果要使每页都有页眉并在页眉中插入图形，可以利用 \fancypagestyle 来改变 plain 页版式的设置。如：

```
\fancypagestyle{plain}{%
\fancyhead{} % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{} % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}}
```

当使用 [twoside] 选项进行双页排版时，仍需将命令

```
\fancyhead[L]{\usebox{\mygraphic}}
```

替换为

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

这样，每一页包括标题页和每章的开始页都会有页眉和图形。

7.6.3 背景中的水印图案

fancyhdr 宏包除了可以用来在页眉和页脚中插入图形之外，还可以用来在每一页面的背景之中插入图形和文本。这对于排版有水印图案的页面是很有用的。

下面的例子将包含在 file.eps 的图形插入每一页面（包括 fancy 版式和 plain 版式）的背景之中：

```
\documentclass{article}
\usepackage{graphicx,fancyhdr}
\newsavebox{\mygraphic}
\sbox{\mygraphic}{%
     \includegraphics[keepaspectratio,height=0.8\textheight,%
                      width=0.8\textwidth]{file.eps}}
\pagestyle{fancy}
\fancyhead{}
\fancyhead[C]{\setlength{\unitlength}{1in}
              \begin{picture}(0,0)
              \put(-2.2,-6){\usebox{\mygraphic}}
              \end{picture}}
\fancypagestyle{plain}{%
     \fancyhead{}%
     \fancyhead[C]{\setlength{\unitlength}{1in}
```

```
                \begin{picture}(0,0)
                \put(2.2,-6){\usebox{\mygraphic}}
                \end{picture}}}
\begin{document}
...
\end{document}
```

这个例子中，图形被放置在页眉中央下方 6in 并偏左 2.2in 的地方。可以通过改变这两个数字来调整图形的位置。因为页眉是在正文之前排版的，所以正文才会在图形的上面不被图形盖住。反之，由于页脚是在正文之后排版的，如果在页脚插入图形，那么位置不合适时图形可能会盖住正文。

如果图形是矢量 EPS 文件，那么可以将其分成两个文件 `file.h` 和 `file.ps`，然后再插入到背景中。这样会使最后得到的 PS 文件小一些。

7.6.4 边空中的图形

在 3.13.2 节中介绍的排版旁注的 `\marginpar` 命令也可以用来将图形放置在边空里。因为旁注的第一行与包含相应的 `\marginpar` 命令的文本行对齐（准确地说，旁注的第一行的参照点在包含相应的 `\marginpar` 命令的文本行的基线上），所以旁注内容太长的话，有可能影响下一个旁注。而且当前页面的旁注不会被排到下一页去，因此，当旁注出现在页面的底部，而底部的边空不够用来排印此旁注时，一部分旁注的内容就会被挤出页面不被排印出来。这种情形常会出现边空排印图形的时候，因为图形通常要占用好几行的位置。

另一方面，因为 `figure` 环境不能在边空中使用，所以不能在边空中放置浮动体，因而 `\caption` 不能在边空中使用。不过可以用 8.7 节介绍的 `\figcaption` 命令来构造不浮动的含标题的图形。例如，在某个句子的后面紧接着输入下面的命令：

```
\newlength{\Lineheight}\settoheight{\Lineheight}{某}
\marginpar{\vspace{-\Lineheight}%
    \begin{minipage}[t]{\marginparwidth}%
    \centering\includegraphics[width=\marginparwidth]{graphics.eps}%
    \figcaption{边空图形的例子}\label{marginparfig}%
    \end{minipage}}
```

就会在某个句子所在文本行的边空中排印一个名为 `graphics.eps` 的小图形。注意，例子中使用了 `\settoheight{\Lineheight}{某}` 来获取当前文本行的高度，而使用命令 `\vspace{-\Lineheight}` 就使得边空图形的顶部与当前文本行的顶部对齐了。

如果要在每一页边空中一个固定的位置都插入同一个图形，仍可以使用 fancyhdr 来达到。例如下面的设置：

```
\documentclass{article}
\usepackage{graphicx,fancyhdr}
\newsavebox{\mygraphic}
\sbox{\mygraphic}{%
     \includegraphics[width=18mm]{graphics.eps}}
\pagestyle{fancy}
\fancyhead[LE]{\setlength{\unitlength}{1mm}
     \begin{picture}(0,0)
     \put(-22,-100){\usebox{\mygraphic}}
     \end{picture}\thepage}
\fancyhead[RO]{\thepage\setlength{\unitlength}{1mm}
     \begin{picture}(0,0)
     \put(2,-100){\usebox{\mygraphic}}
     \end{picture}}
\fancypagestyle{plain}{%
     \fancyhead[LE]{\setlength{\unitlength}{1mm}
     \begin{picture}(0,0)
     \put(-22,-100){\usebox{\mygraphic}}
     \end{picture}\thepage}
\fancyhead[RO]{\thepage\setlength{\unitlength}{1mm}
     \begin{picture}(0,0)
     \put(2,-100){\usebox{\mygraphic}}
     \end{picture}}}
\begin{document}
...
\end{document}
```

是一个双页排版的设置，它会将名为 graphics.eps 的图形放置在每一页（无论 fancy 页版式还是 plain 页版式）距页眉 10cm 的外侧边空中。注意，边空的宽应设置地足够大使得能放下所要插入的图形。

7.7 图形文本的替换

目前有许多画图软件或程序都能输出 EPS 格式的文件，但是大多数都不支持科学文献中常使用的一些特殊符号如数学符号和公式。另一方面，目前的一些基于 LaTeX 的画图程序功能都不够强，使用起来也不是很方便。为了能在示例图中方便地插入所需的特殊符号或公式，这里介绍两个可以用来替换图形中文本的宏包。

7.7.1 Psfrag 宏包

Psfrag 是一个易于使用的用来替换图形中文本的宏包。用它可以将图形中的文本替换为任何 LaTeX 结构。可以按下面 3 个步骤来使用这个宏包：

1) 在 LaTeX 源文件的导言部分插入命令 `\usepackage{psfrag}`。

2) 在正文中使用 `\psfrag` 命令并指明用来替换的文本和将被替换的文本。这条命令会将其后所有用 `\includegraphics` 插入的图形中相应的文本都替换掉。

3) 按正常方式使用 `\includegraphics` 插入图形。

`\psfrag` 命令的语法是：

`\psfrag{`*PStext*`}[`*posn*`][`*PSposn*`][`*scale*`][`*rot*`]{`*text*`}`

PStext 是将要被替换的图形中的文本。

posn 是一个可选项，表示相对于新 LaTeX 文本参照点的放置位置，其值可取图 7.1 中指出的 12 种字母组合之一，默认值是 `Bl`，即左下角。

PSposn 是一个可选项，表示相对于图形中将被替换的 LaTeX 文本的参照点的位置。可取的值与 *posn* 一样，默认值也是 `Bl`。

scale 可选的伸缩因子，表示将文本伸缩一个倍数之后再用来替换旧文本，默认值是 1。为了得到最佳结果，尽量避免使用伸缩因子，必要时可使用 `\small`、`\large` 等命令。

rot 可选项，默认值为零。表示新文本相对于旧文本的旋转角度，正值表示逆时针旋转。这个选项特别有用，因为有些画图程序只允许水平放置文本。

text 用来替换图形中旧文本的新文本。数学符号或公式必须放在一对美元符号中。如 `$\frac12$`。

下面是一个结合 `\colorbox` 来使用 Psfrag 宏包的例子：

```
\psfrag{q1}[][]{\colorbox{white}{$q_1$}}
\psfrag{base}{\fcolorbox{black}{white}{Base}}
\psfrag{Actuator}[l][l]{\shortstack{Hydraulic\\ Actuator}}
\includegraphics{mass.eps}
```

图 7.3 是一个文件名为 mass.eps 的 EPS 格式的原始图形，图 7.4 利用了 Psfrag 宏包将图 7.3 中某些文字替换成另外的文字或盒子。

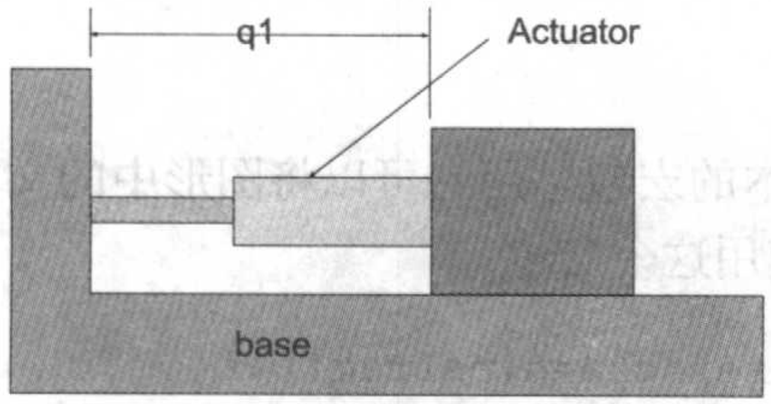


图 7.3 未使用 Psfrag 时的原图

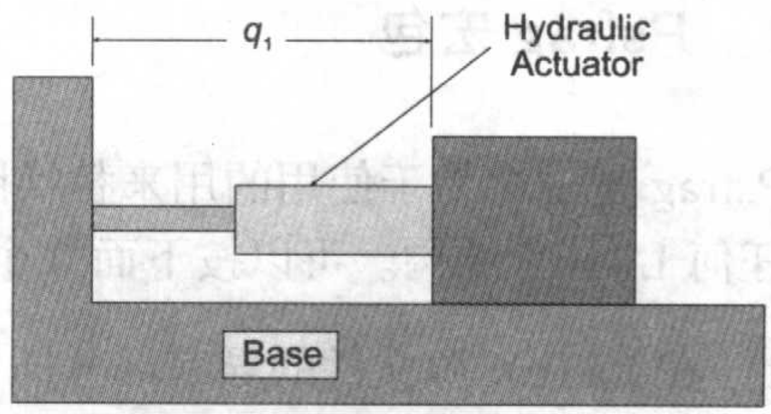


图 7.4 使用 Psfrag 后的图形

7.7.2 Overpic 宏包

虽然 Psfrag 宏包的功能很强大，使用也很方便，但是它却与某些程序生成的 EPS 图形不兼容。这使得 Psfrag 宏包的使用受到一定的限制。本节再介绍一个可用来替换 EPS 图形之中的文本的 overpic 宏包。这个宏包将 LaTeX 的 picture 环境与图形宏包套件中 \includegraphics 命令结合起来，能将一个图形直接放置到另一个图形上面。也可以在一个图形四周加上带刻度的标尺，使得图形中图形元素的位置一目了然，因此可以很方便地将各种 LaTeX 结构放到图形中的指定位置。

要使用 overpic 宏包，首先需在 LaTeX 源文件的导言部分插入命令：

\usepackage[*option*]{*overpic*}

其中 *option* 是 overpic 宏包选项，它有如下三种选择：

percent 这是默认选项。表示图形标尺的高度和宽度两者之中较大者是 100，另一个由图形的宽高比例得到。

permil 表示图形标尺的高度和宽度两者之中较大者是 1000，另一个则由图形的宽高比例得到。

abs 表示图形标尺的高度和宽度是绝对的，即由单位长度决定。

然后，在正文的插图处使用宏包提供的 overpic 环境：

```
\begin{overpic}[选项]{图形名}
LaTeX picture 环境中的图形元素
\end{overpic}
```

其中图形名是所插入的 EPS 图形的文件名，而选项则可以有如下几种选择：

scale 图形的伸缩因子，默认值为 1。

grid 此选项表示在图形四周加标尺。

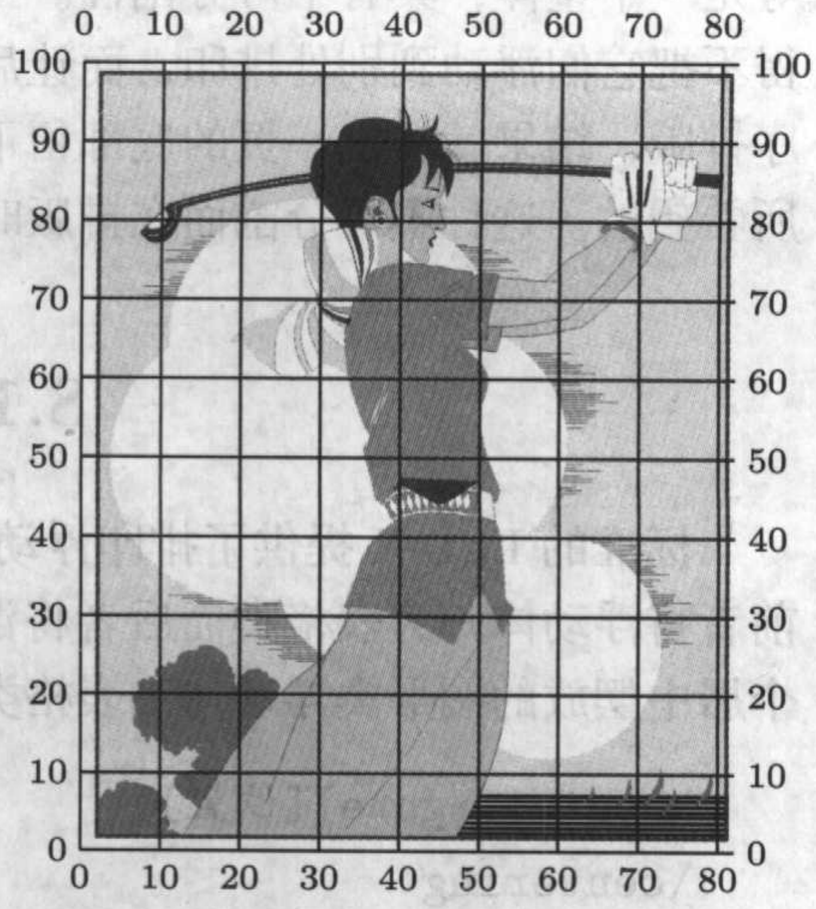
tics 标尺中一个刻度的值，对于 *abs* 和 *percent* 来说其默认值是 10，对于 permil 来说其默认值是 100。

unit 画图环境的单位长度，与 \unitlength 一样。对于 *percent* 和 *permil* 来说已被分别内定为 1/100 和 1/1000。但是对于 *abs* 来说，*unit* 的值必须指明，或在环境前面用 \unitlength 设置。

注意，两个选项之间必须用逗号隔开，且不含空格。环境中的图形元素可以是一些在 LaTeX picture 环境中使用的任何画图命令。下面是宏包中提供的例子：

```
\begin{overpic}[scale=.23,%
                grid,tics=10]%
               {golfer.eps}
\end{overpic}
```

将图形按比例缩小到原来的 0.23 倍插入正文中，并在图形四周加上标尺，纵向和横向标尺中较长者为 100，标尺中一个刻度的值是 10。




```
\begin{overpic}[scale=.23]%
               {golfer.eps}
\put(5,45){\huge LaTeX}\put(55,10)%
{\includegraphics[scale=.07]%
                 {golfer.eps}}
\end{overpic}
```

先插入一个缩小到原来的 0.23 倍的图形，然后再将原图缩小到原来的 0.07 倍并插入到（相对于第一次插入的图）坐标为 (55,10) 的位置，同时将“LaTeX”插入到 (5,45) 的位置。

第8章　浮　动　体

如果文稿的所有内容都能按照作者设计的顺序进行排版，那么文稿就很容易阅读。但在排版中这是很难做到的，或者说从技术上是不可能的。因为某些素材如表格、图形等是一个整体，具有不可分割性。当某一页底部的空间已不够排印这种素材时，系统不得不把它们浮动到别处排印，而让后面的文字接着排下去。假如我们强迫系统按自然顺序排版，结果就在这一页的底部留下一大片空白，从而影响文稿的美观，这也不是我们所希望的。这种不可分割而在排版时有可能浮动到别处排印的整体素材称为浮动体。

8.1　创建浮动体

标准的 LaTeX 提供了排印浮动体的两种环境，它们是 table 环境和 figure 环境。前者将浮动体归为表格类而后者将浮动体归为图形类。比如用户可用下面的方法创建一个居中摆放的图形类浮动体和表格类浮动体：

```
\begin{figure}[位置]
\centering
此处是放置图形的命令
\caption{标题}\label{fig:graph}
\end{figure}

\begin{table}[位置]
\centering
此处是画表格的命令
\caption{标题}\label{tab:table}
\end{table}
```

其中 \caption 命令用来排印浮动体的标题。图形浮动体和表格浮动体的标题会分别进入图形目录和表格目录中，使用命令 \listoffigures 和 \listoftables 可以将这两个目录排印出来。命令 \label 紧跟在标题后面贴上一个标签，使用户能交叉引用这个标题。可选参数位置允许用户来指示浮动体有可能被放置的位置，它可以取如下几个字母的组合（默认值是 tbp）：

- h　当前位置。将浮动体放置在浮动体环境出现的地方。如果当前位置放不下，那么此参数不起作用。
- t　页面顶部。将浮动体放置在浮动体环境所在页的顶部。

b 页面底部。将浮动体放置在浮动体环境所在页的底部。

p 浮动页。将浮动体放置在只允许包含浮动体的页面上。

! 放在参数的最前面时，将取消对浮动体的大多数限制。

位置参数中字母的组合顺序并不影响浮动体的放置，事实上 LaTeX 总是按照 h、t、b、p 的顺序来放置。给出的参数越多 LaTeX 越容易处理，如果只给出一个参数，那么一旦所指定的位置不能放置，浮动体就会被搁置起来，从而阻挡了后面同类浮动体的放置。LaTeX 最多只允许有 18 个浮动体被阻塞，超过这个数时，系统就产生“Too Many Unprocessed Floads”的错误。

LaTeX 在处理浮动体时会遵循以下原则：

1) 只将浮动体放置在位置参数中指定的地方。

2) 浮动体只能被放在当前页或者后面的页，而不能浮动到前面的页。

3) 同一类的浮动体是按顺序放置的，且只有当前面的浮动体放置好之后，才放置当前浮动体。一个被搁置的浮动体会阻塞后面的同类浮动体。

4) 浮动体的尺寸不能超过版心的尺寸，否则系统产生“overfull page”的错误。

另外，LaTeX 在处理浮动体时还遵循一些影响文稿美观的条件。例如，一页上的浮动体总数不会超过由计数器 totalnumber 所规定的数。

在双栏排版的页面中，如果要使浮动体跨越两栏并置于页面的中间，那么就必须使用带星号的浮动体环境命令，即

```
\begin{figure*}[位置]  图形  \end{figure*}
\begin{table*}[位置]  表格  \end{table*}
```

注意在这两个环境中，位置选项不能取 h 和 b。

8.2 控制浮动体的参数

为了得到专业排版效果，有必要进一步了解 LaTeX 是如何处理浮动体的。下面列出了控制浮动体的参数。必要时可以在正文中修改其中的某些参数值，这种修改只对其后的页面起作用。当然如果在导言部分修改了这些参数的值，那么它就对整个文稿起作用了。

下面是 4 个用于控制浮动体的计数器：

topnumber 这是一个计数器（默认值是 2），指示在一页的顶部最多允许放置多少个浮动体。可以用 \setcounter 命令（见 2.13 节）修改它的值。

bottomnumber 这是一个计数器（默认值是 1），指示在一页的底部最多允许

放置几个浮动体。可以用 \setcounter 命令修改它的值。

totalnumber 这也是一个计数器（默认值为 3），表示一页中允许放置浮动体的最大数目。可以用 \setcounter 命令修改它的值。

dbltopnumber 这是一个计数器，其作用与 topnumber 类似，只是用于控制双栏排版的页面中的浮动体。默认值为 2。

下面是控制浮动体的参数：

\topfraction 置于页面顶部的浮动体的高度与页面高度的最大比值，默认值是 0.7，即顶部浮动体的高度不能超过页面高度的 70%。但当一页中有多个含选项 [t] 的浮动体时，则这些浮动体高度的和不能超过整个页面高度的 60%，否则这些浮动体一个也不会被放置。这个参数的值可以用 \renewcommand 命令来改变。

\bottomfraction 置于页面底部的浮动体的高度与页面高度的最大比值，默认值是 0.3，即只有高度不超过页面高度 30%的浮动体才能放在页面底部。可以用 \renewcommand 来改变它的值。

\textfraction 页面中文本占整个页面的最小比例，默认值为 0.2，即浮动体占整个页面的比例不能超过 80%。可以用 \renewcommand 来改变它的值。

\floatpagefraction 浮动页（仅含有浮动体的页面）中浮动体占整个页面的最小比例。默认值为 0.5，这使得浮动页中空白所占的比例不能超过 50% 以上。可以用 \renewcommand 来改变它的值。

\floatsep 表示页面顶部或底部浮动体之间的垂直弹性距离。对于 10pt 和 11pt 字体大小的文稿，这个参数的默认值是 12pt plus 2pt minus 2pt；但对于 12pt 字体大小的文稿，这个参数的默认值是 14pt plus 2pt minus 4pt。可以用 \setlength 命令来改变这个参数的值。

\textfloatsep 表示页面顶部或底部浮动体到文本之间的垂直弹性距离。默认值为 20pt plus 2pt minus 4pt。可以用 \setlength 命令来改变它的值。

\intextsep 出现在文本中间的浮动体与上下方文本之间的垂直弹性距离。默认值与 \floatsep 一样。可以用 \setlength 命令来改变它的值。

\topfigrule 是一条在页面顶部最后一个浮动体之后，而在 \textfloatsep 之前被执行的命令，但这条命令不允许在浮动体与文本之间产生额外的垂直距离。通过重新定义这条命令可以在浮动体与

文本之间画水平直线或者放置其他的东西。由于这条命令原先的奇特定义方法，使得对它的重新定义只能用 \newcommand 命令，而不是用 \renewcommand 命令。注意此命令对含选项 h 被置于当前位置的浮动体无效，即使它正好出现在页面的顶部。

\botfigrule 这条命令与 \topfigrule 类似，但它在 \textfloatsep 之后，而在底部第一个浮动体之前被执行。

\dblfigrule 用于双栏排版中，作用与 \topfigrule 类似。

下面的参数用于控制双面排版中的浮动体：

\dbltopfraction 此参数用于双栏排版的页面中，其作用与 \topfraction 类似，默认值为 0.7。可以用 \renewcommand 来改变它的值。

\dblfloatpagefraction 用于双栏排版的页面中，作用与 \floatpagefraction 类似，默认值为 0.5。

\dblfloatsep 这个参数用于双栏排版的页面中，其作用以及默认值都与 \floatsep 类似。

\dbltextfloatsep 此参数用于双栏排版的页面中，它定义了一个弹性距离，作用与 \textfloatsep 类似，其默认值也与 \textfloatsep 的默认值相同。

修改上面所列出的这些参数的值就会改变 LaTeX 系统控制浮动体的算法，从而会影响浮动体的放置地点。为了得到美观的排版结果，用户在修改这些参数的值时必须非常小心，特别要注意参数之间的相互依赖关系，否则有可能出现低劣的排版结果。

如果在一个包含许多浮动体的文稿中只使用默认的参数值，那么排版结果中可能会出现几个浮动页，也就是只含有浮动体的页面。浮动页中通常会有许多空白，例如可能出现只包含一个浮动体而在下方留下几乎半页空白的浮动页。之所以如此，是因为 LaTeX 系统的算法会尽量将等待放置的浮动体放到每一页的后面，因而也就产生了尽可能多的浮动页直到没有浮动体为止。用来控制浮动页的参数 \floatpagefraction 指明浮动页中浮动体占页面的最小比例，默认值是页面的一半。而在标准的设置中浮动体的选项是 tbp，这意味着每一个尺寸稍大于半页的浮动体都允许被放到浮动页中。于是，增大这个参数的值可以避免出现浮动页中有一半空白的现象。

然而增大参数 \floatpagefraction 的值也使浮动页的产生变得更困难。其结果是某些浮动体可能被搁置起来不作处理，这就阻止了对后面的浮动体的放置。为避免这种情况，最好对那些产生问题的浮动体，明确给出摆放位置，比如使用选项 tb。

下面是对修改浮动体参数值的一些建议和应遵守的规则：

1) 不要使 \textfraction 的值小于 0.15，这会导致难以阅读的文本页面。如果浮

动体的高度超过页面的 85%，把它放到单独一个浮动页面中，效果会更好些。

2) 不要让 \topfraction 的值小于 1−\textfraction，这会使 LaTeX 的浮动定位算法发生冲突。

3) 不要在页面的底部放置太多的浮动体，因而应把 \bottomfraction 的值设置得小一些，最好比 \topfraction 的值小。但不要把它的值设置成 0，这也会使 LaTeX 的浮动定位算法发生冲突。

4) 为了不使浮动体被搁置起来，\floatpagefraction 的值应与 \topfraction 的值协调。一般情况下，如果浮动体选项中含有 tp，则应满足不等式：

\floatpagefraction⩽\topfraction -0.05。

同样浮动体选项中含有 bp 时，则应满足不等式：

\floatpagefraction⩽\bottomfraction -0.05。

需要注意的是默认值并不满足这个不等式，所以在处理含 bp 选项的浮动体时，可能会出现一些问题。

8.3 限制浮动位置

从页面美观出发，LaTeX 的浮动定位算法总是优先考虑将浮动体放置在页面顶部，但这种情况并不总是可以接受的。比如某一节内容从页面的中间开始，恰好这一节一开始就有一个浮动体，此时由于这个浮动体被置于页面的顶部，这使得它看起来仿佛是上一节内容的一部分。鉴于此，就有必要将这种浮动体放到后面去。当然用户可以使用 flafter 宏包，它使得所有浮动体不能被放置在对应的浮动环境所在位置的前面。注意，这个宏包是对一切浮动体都起作用的。有时我们更喜欢一种局部的解决办法。LaTeX2e 提供了一条命令 \suppressfloats，用它可以阻止浮动体出现在当前页面的顶部和底部。这条命令还可以带选项 t 或者 b，其具体意义如下：

\suppressfloats[t]	它后面的浮动体不能放在当前页的顶部。
\suppressfloats[b]	它后面的浮动体不能放在当前页的底部。
\suppressfloats	它后面的浮动体不能放在当前页的顶部和底部。

另外一种影响浮动体位置的方法是在浮动体环境的选项中使用感叹号，如 [!b]。这种选项中带感叹号的浮动体就会不顾上节中介绍的那些浮动体参数（除了浮动页的参数）对它的限制，而将浮动体放置在选项中提供的位置，只要满足下面两个条件：

1) 浮动体要能放进页面中，即浮动体的的高度不超过页面的高度 \textheigth。

2) 前面没有被搁置的同类浮动体。

8.4 处理被搁置的浮动体

尽管 LaTeX 的浮动定位算法在通常情况下能很好地处理浮动体，但有时仍会觉得找不到合适的位置来放置某些浮动体。此时 LaTeX 系统就会将它们暂时保存起来不作处理，以等待合适的位置。这种已被 LaTeX 读入内存而未作处理的浮动体称为被搁置的浮动体。由于 LaTeX 对被搁置的浮动体的总数是有限制的（不超过 18 个），在有大量浮动体的文稿中就有必要强迫 LaTeX 系统来处理被搁置的浮动体。

最简单的处理被搁置的浮动体的方法是使用命令：

`\clearpage`

它强迫 LaTeX 立即处理那些已被搁置的浮动体，并且开始一个新页。这条命令虽然很有效，但也可能导致某些页面的下方出现大片空白。

对于大多数情况来说，最好的办法是使用 placeins 宏包提供的命令：

`\FloatBarrier`

同 `\clearpage` 一样它也强迫 LaTeX 立即处理那些被搁置的浮动体，但并不开始一个新页。如果想在每一节开始前就把被搁置的浮动体排印出来，那么可以在调用 placeins 宏包时使用 `section` 选项，即

```
\usepackage[section]{placeins}
```

它实际上重新定义了排版节标题的命令 `\section`，使得在每个 `section` 的前面都加上一个 `\FloatBarrier`。此时当一个 `section` 从某页的中间开始时，前一个 `section` 中的浮动体就不会被放在这一页的底部。假设始终允许浮动体被放在底部的话，那么可以用另外一个选项 `below`，即

```
\usepackage[below]{placeins}
```

这样，即使一个 `section` 从某页的中间开始，前一个 `section` 中的浮动体也可能会被放在这一页的底部。

另外一种办法是使用 afterpage 宏包中提供的命令

`\afterpage{`*commands*`}`

其中的 *commands* 是 LaTeX 的一些命令。在上面这条命令之后的第一个自然分页前面，这些 *commands* 就会被执行。因而命令

```
\afterpage{\clearpage}
```

就会在它本身所在的页结束时把被搁置的浮动体排印出来。当文稿中有很多浮动体时，

此命令出现之前被搁置的浮动体可能就已超过了限制，此时就需要在多处使用这条命令。还要注意的是此命令不能用于多栏排版的文稿中。

8.5 浮动体的标题

要对浮动体的内容进行简短说明，通常是使用下面的标题命令：

```
\caption[短标题]{主标题}
```

这条命令只有在浮动体环境中才能使用，其中的可选项短标题会进入图形目录或表格目录中。如果省略短标题，那么主标题将会被写入那些目录中。当主标题的内容多于一行时，最好附加一个短标题，否则，图形目录或者表格目录会排列的很不整齐，从而难以阅读。

在排版中图形类浮动体的标题出现的形式类似于 Figure 3.2: 主标题,而表格类浮动体标题出现的形式类似于 Table 3.2: 主标题。其中 Figure 和 Table 是标题名称（也称标题标记），它们分别由命令

```
\figurename 和 \tablename
```

来定义，因而可以用 \renewcommand 改变它们的形式。

事实上，每个 \caption 命令都会调用 LaTeX 内部命令

```
\@makecaption{numb}{text}
```

其中 *numb* 是 LaTeX 根据浮动体类型计数器自动产生的序号，*text* 代表浮动体的内容。这条产生标题的命令的定义是：

```
\newcommand{\@makecaption}[2]{% #1 is Figure 1, #2 is caption text
    \vspace{10pt}\sbox{\tempbox}{#1: #2}%
    \ifthenelse{\lengthtest{\wd\tempbox > \linewidth}}%
        { #1: #2\par}% More then one line
        {\begin{center}#1: #2\end{center}}}
```

我们看到在留出一个 10pt 的垂直空白之后，标题被存入一个名为 \tempbox 的临时盒子中，然后比较标题的宽度与行宽 \linewidth 的大小。标题内容不超过一行时，标题是被居中排版的；如果标题内容超过一行，那么标题就被当作一个与页面同样宽度的不缩格的段落进行排版。

按照这种方法重新定义 \@makecaption 就可以得到不同样式的标题。当然，这对于初学者是有些困难的。如果使用 caption2 宏包就可以很方便地控制标题的样

式。caption2 宏包的使用方法是在源文件的导言部分输入如下命令：

```
\usepackage[选项]{caption2}
```

其中选项可以是 center，flushleft，flushright，用来使标题每行居中、左对齐和右对齐；可以是 scriptsize，footnotesize，small，normalsize，large 和 Large，用来调整标题主体文字的大小；也可以是 up，it，sl，sc，md，bf，rm，sf 和 tt，用来改变标题名称的字体。另外还可以用下面这些选项：

normal	这是默认选项。这个形式的标题文本两边对齐，最后一行左对齐。
hang 或 isu	标题文本从第二行开始缩进与第一行中标题内容的第一个字对齐。
indent	这个选项与 normal 相似，只是标题文本从第二行开始向里缩进一个由 \captionindent 定义的长度。这个长度的默认定义是 0pt，可以用 \setlength 命令对它重新赋值。
centerlast	标题文本两边对齐，最后一行居中。
nooneline	标题只有一行时，也看成多行来对待。因而标题可以不居中。
ruled	支持 float 宏包提供的带边框的浮动体。

caption2 宏包中还定义了如下一些命令用来改变所有或者个别浮动体标题的形式：

\captionstyle{选项}	用来改变标题的样式。
\captionfont	标题字体命令，可以用 \renewcommand 重新设置它的属性。
\captionlabelfont	标题名称字体命令，可以用 \renewcommand 重新设置它的属性。
\setcaptionwidth{*width*}	设置标题宽度为*width*。
\setcaptionmargin{*mar*}	设置标题到两边页面边界的距离为 *mar*。
\captionlabeldelim	设置标题命令与标题内容之间的分隔符，默认值为冒号，可用 \renewcommand 来改变。
\onelinecaptiontrue	单行标题居中排版。
\onelinecaptionfalse	重新设置单行标题的形式。

根据命令 \caption 所在位置的不同，标题可以出现在浮动体的上方也可以出现在浮动体的下方。\caption 命令在浮动体之前，则标题放在浮动体上面；若 \caption 命令在浮动体之后，则标题放在浮动体下面。标题到上下方文本之间的距离分别由参数 \abovecaptionskip 和 \belowcaptionskip 来控制。

有时也可能出现把浮动体的标题放在旁边的情况，这可以通过把 \caption 命令和浮动体分别放在不同的小页环境来实现。例如，要把一个图形的标题放在图形的左侧，可以用如下方法得到：

```
\begin{figure}
\centering
\begin{minipage}[c]{.45\textwidth}
\centering \caption{左侧标题} \label{fig:side-cap}
\end{minipage}%
\begin{minipage}[c]{.45\textwidth}
\centering \includegraphics[width=\textwidth]{graphics.eps}
\end{minipage}
\end{figure}
```

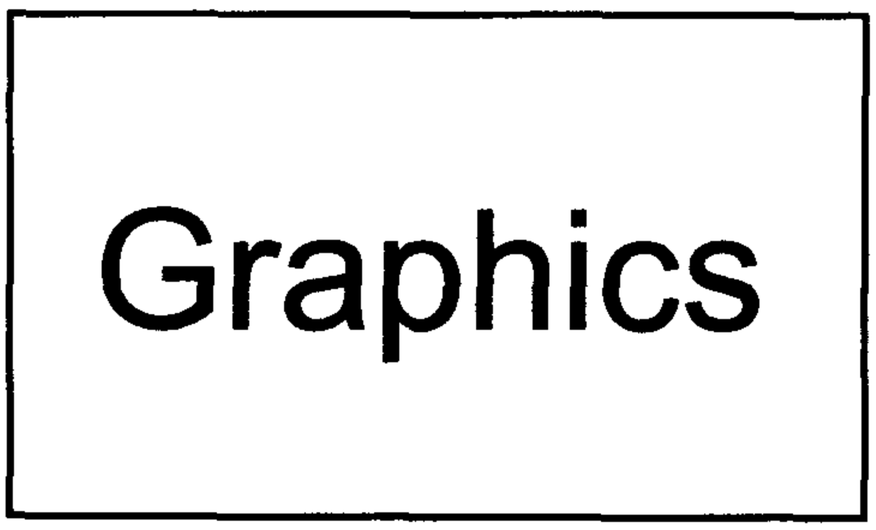


图 8.1 左侧标题

如果希望更灵活更方便地控制标题在浮动体两边的位置，可以使用 sidecap 宏包。这个宏包定义了两个环境 SCfigure 和 SCtable，分别用来将标题放在图形和表格的旁边。标题到包含图形或表格的小页的距离可以用长度参数 \sidecaptionsep 来定义。要使用这个宏包只需在源文件的导言部分调用命令

```
\usepackage[选项]{sidecap}
```

其中选项可以取：

outercaption 这是默认值，表示标题放在左页的左侧，右页的右侧。

innercaption 表示在左页中标题放在图形的右侧，而在右页中标题放在图形的左侧。

leftcaption 表示标题总是放在图形的左侧。

rightcaption 表示标题总是放在图形的右侧。

raggedleft，raggedright，ragged 这三个选项能更好地调整标题在包含它的小页中的位置。raggedleft 意指标题在小页中居右，而 raggedright 表示居左。

当在正文部分插入图形类或表格类浮动体时，可以使用下面的环境：

```
\begin{SCfigure}[相对宽度][位置]··· \end{SCfigure}
\begin{SCtable}[相对宽度][位置]··· \end{SCtable}
\begin{SCfigure*}[相对宽度][位置]··· \end{SCfigure*}
\begin{SCtable*}[相对宽度][位置]··· \end{SCtable*}
```

其中相对宽度是指标题相对于浮动体的宽度，默认值是 1.0。默认值可以通过重新定义参数 \sidecaptionrelwidth 的值来改变。一个较大的相对宽度值（如 50）可以使标题占用尽可能大的宽度。位置就是一般浮动体的浮动位置选项，可以取 t，b，p，！及它们的组合。带星号的环境可以用来在多栏排版时使浮动体跨栏排版。

8.6 创建新的浮动体类型

除了图形和表格之外，如果用户还想将其他的某些素材作为浮动体来对待，而又不想把它们归为图形类或表格类，那么就有必要定义新的浮动体类型。利用 float 宏包中的命令

```
\newfloat{type}{placement}{ext}[within]
```

就可以定义一个新的浮动体类型。这条命令有四个参数，前三个是必需的，而最后一个是可省略参数，它们的意义如下：

type 是定义的浮动体类型，比如 table、figure、program 等。如果类型定义为 program，就可以用 \begin{program} 来创建这种类型的浮动体。

placement 是所定义的浮动体的默认位置。与标准的 LaTeX 一样，可以选 t, b, p 和 h（以及这些字母的组合）分别表示顶部、底部、浮动页和当前位置。在创建个别浮动体时，还有一个特别的位置选项 H，它指示将浮动体放在当前位置并且不能放在别处。

ext 产生辅助文件的扩展名。这个辅助文件记录了此类浮动体的标题，当排印此类浮动体的目录时，系统会从这个文件读取信息。

within 这是一个可省略参数，表示章节名，如 chapter, section 等，它决定是否将此类浮动体在某种章节中独立编号。例如，当此参数是 chapter 时，则这一类的浮动体就在每一章中按独立顺序编号。

下面这条命令

```
\floatstyle{style}
```

用来设置浮动体的式样，其中 *style* 是式样名，它可以选下面三种形式之一：

`plain` 这是标准 LaTeX 中浮动体式样，没有什么特别之处，惟一不同的是它的标题被排印在浮动体的下方，而不管 `\caption` 这条命令是在上面还是在下面。

`boxed` 在这种式样中，浮动体被置于一个方框中，标题在方框的下面。

`ruled` 这种浮动体式样将浮动体标题置于浮动体上方两条横线中，而且浮动体下方有一条横线，作为浮动体的结尾。

此命令只设置它后面所有由 `\newfloat` 定义的浮动体式样，直到另一个 `\floatstyle` 出现为止。它对标准 LaTeX 中已有的 `figure` 和 `table` 浮动体不起作用，但是可以用命令

`\restylefloat{`*type*`}`

来改变这两个浮动体的式样。例如可以用命令

```
\floatstyle{ruled}
\restylefloat{table}
```

将表格浮动体的式样设置为 `ruled`。类似于 Figure, Table 这样的浮动体标题名称可以用命令

`\floatname{`*type*`}{`名称`}`

来定义。例如，命令 `\floatname{program}{Program}` 将 `program` 类型的浮动体的名称定义为 Program。在默认的情况下，如果不给出标题名称的定义，那么由 `\newfloat` 定义的浮动体类型就将它的类型参数 *type* 作为这种类型的浮动体的标题名称。下面的命令

`\floatplacement{`*type*`}{`位置`}`

用来改变所给类型的浮动体的默认放置位置。例如，`\floatplacement{figure}{tp}` 将图形浮动体的默认位置设置为 `tp`。如果要建立某种类型的浮动体的目录，可以使用命令

`\listof{`*type*`}{`*title*`}`

这里 *type* 就是要建立目录的浮动体类型名，*title* 是所建立的目录的题目。这条命令类似于标准 LaTeX 中的命令 `\listoffigures` 和 `\listoftables`。

8.7 不浮动的图形和表格

通常情况下，我们总是使用浮动的图形和表格来增强排版效果，但偶尔也会希望某些个别的图形和表格不要浮动就放在源文件中所出现的位置。当然我们可以使用 float 宏包，并按照下面的方法

```
\restylefloat{figure}
\begin{figure}[H]
图形命令
\end{figure}
\restylefloat{table}
\begin{table}[H]
表格命令
\end{table}
```

来排版不浮动的图形和表格。不过使用 float 宏包的 H 选项也会有下面的问题：

1) 如果当前页没有足够的空间来放置带 H 选项的图形或表格，则它们会被放置在下一页的顶部。但是，当前页有脚注时，脚注会紧接着文本后面排出，而不是像通常那样排在脚注的位置。此时需要在文本后面加上足够的垂直空白距离，使脚注移到正常的位置。

2) 我们有时希望浮动体的标题放在浮动体的上方或者旁边，但使用 float 宏包定义的 plain 类型的浮动体总是将标题放在浮动体的下方，而不管命令 \caption 出现在哪个位置。

下面介绍一个更灵活的方法来排版不浮动的图形和表格。事实上，\caption 命令之所以可以在 figure 和 table 环境中使用，是因为这两个环境中都定义了内部命令 \@captype。因此，通过定义这条命令也可以在 figure 和 table 环境之外使用 \caption 命令来获得图形和表格的标题，并且用这种方法得到的标题也会进入图形或者表格目录中。由于命令 \@captype 中含有字符 @, 在源文件中对其定义时必须把定义夹在 \makeatletter 和 \makeatother 之中。例如，下面的命令

```
\includegraphics{file.eps}
\makeatletter\def\@captype{figure}\makeatother
\caption{This is the caption}
```

会排印出一幅名为 file.eps 的图，并且带有一个标题。将这种定义放在源文件的导言部分用起来会更方便些。例如，在导言部分用下面的方法

```
\makeatletter
\newcommand{\figcaption}{\def\@captype{figure}\caption}
\newcommand{\tabcaption}{\def\@captype{table}\caption}
```

\makeatother

定义命令 \figcaption 和 \tabcaption。这样在正文中不管是否在 figure 环境中都可以使用 \figcaption 命令来排版图形标题。同样即使是在 table 环境之外也可以使用 \tabcaption 命令来得到表格标题。

8.8 几种不同的浮动环境

8.8.1 用 floatflt 宏包处理宽度窄小的浮动体

用标准 LaTeX 的 figure 环境和 table 环境来排版那些具有窄小宽度的图形或表格时，由于图形或表格被置于中间，两边就会出现较多的空白，显得与页面不太协调。为此，floatflt 宏包（floatfig 宏包的扩展）提供了 floatingfigure 环境和 floatingtable 环境。环境中的图形或表格根据需要可以被置于页面的左边或右边，并且在它们的旁边可以排版正常文本内容，如图 8.2 所示。要使用这个宏包，首先需在源文件的导言部分输入下面的命令：

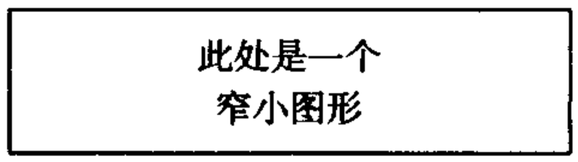


图 8.2 floatflt 宏包例子

\usepackage[*option*]{floatflt}

其中 *option* 是选项，它可以取为

rflt 图形或表格在位置参数默认时被放置在段落的右边。

lflt 图形或表格在位置参数默认时被放置在段落的左边。

vflt 图形或表格在位置参数默认时被放置在奇数页段落的右边或偶数页段落的左边。这也是 *option* 的默认值。

在文本的主体部分可以用如下方法来排版一幅窄小宽度的图形：

\begin{floatingfigure}[*option*]{*width*} 图形命令 \end{floatingfigure}

这里 *width* 是宽度，其值必须比图形的实际宽度大一些，*option* 是当前图形的位置选项，可以取如下一些字母：

r 将当前图形放置在段落的右边。

l 将当前图形放置在段落的左边。

p 将当前图形放置在奇数页段落的右边或偶数页段落的左边。

v 用上面宏包中的位置选项作为此处的位置选项。若宏包中无选项，则将当前图形放置在奇数页段落的右边或偶数页段落的左边。

要排版窄小宽度的表格可以用下面方法：

```
\begin{floatingtable}[option]{
\begin{tabular}{列格式}
此处输入表格内容
\end{tabular}}
\caption{标题内容}
\end{floatingtable}
```

其中 *option* 是位置选项，可以取字母 r，l，p，v 之一，意义与 floatingfigure 的选项的意义完全相同。注意，这里没有宽度选项，它实际上是以表格的宽度为宽度。因此，\begin{tabular} 前面的左花括号“{”和 \end{tabular} 后面的右花括号“}”是不能省略的。例如 50 页的表 3.2 就使用了带选项 r 的 floatingtable 环境。

floatingtable 环境中的 tabular 环境可以被其他的一些命令（如 \parbox）代替，但在这个环境中不能使用 tabbing 环境。

floatingfigure 环境和 floatingtable 环境可用于多栏排版的页面中但只能用在垂直模式中，即两个段落之间。这类命令一旦出现，LaTeX 系统会立即对之进行处理。也就是说 LaTeX 系统会立即检验当前页是否还有垂直空白用来排版这类图形或表格。如果没有足够的垂直空白，这类图形或表格就会被排到下一栏或下一页。

8.8.2 用 wrapfig 宏包对图形绕排

另外一个可以用来处理窄小图形或表格的宏包是 wrapfig。此宏包提供了图形浮动体环境 wrapfigure 和表格浮动体环境 wraptable，用来处理那些具有窄小宽度的图形和表格，使得图形或表格排在文本的一侧，文本在图形或表格的一边绕排。wrapfigure 环境的用法是：

```
\begin{wrapfigure}[行数]{位置}[超出长度]{宽度}
图形命令
\end{wrapfigure}
```

（wraptable 环境的用法完全相同）其中的参数意义如下：

行数　是图形高度所占的行数，默认时 wrapfig 宏包会根据图形的高度自动计算。文本在图形的周围绕排，并在图形的顶部和底部留出长度为 \intextsep 的垂直空白，在图形的旁边留出长度为 \columnsep 的水平空白。

位置　此选项决定图形是放在文本的左边还是右边，它必须是下面这些字母之一：

r，R　表示把图形放在文本的右边。

l，L　表示把图形放在文本的左边。

i，I　在双面排版的文稿中，将图形放在靠近装订线的一侧。

o，O　在双面排版的文稿中，将图形放在离装订线远的那一侧。

小写字母表示将图形放置在当前位置，而大写字母则表示图形允许被浮动到别处。但这种浮动是有很多限制的，最好的排版效果是通过手工调整来明确给出图形的位置。不过在修改稿件的过程中，允许浮动是有好处的，因为任何手工的调整都有可能被一次修改破坏掉。一般是在打印文稿前，再来做最后的手工调整。

超出长度　是指图形超出边界的长度，默认值是 0pt，此默认值是由长度参数命令 \wrapoverhang 定义的，因此可以用 \setlength 命令对其重新赋值。超出长度也可以取一个特殊的长度单位，即 \width，其意义是图形的宽度。例如，[0.5\width] 表示将图形的中间置于页面的边界上，[\width] 表示将整个图形都放入到边缘的空白中。\width 是图形的实际宽度，因此可能要比设置的宽度更大一些。

宽度　是指图形的宽度。事实上，LaTeX 会将图形放入一个宽度为宽度的盒子中。因此，当图形的实际宽度比所设置的宽度大时就出现“overfull hbox”的警告。不过图形的标题总是按照所给的宽度来排版。然而，如果将宽度设置为 0pt，那么 wrapfig 宏包会自动计算图形的实际宽度，此时就不能排版标题，这是因为已将宽度设置为零。

另外，在使用 wrapfigure 环境和 wraptable 环境时还应注意如下事项：

1) 环境不能置于一个正好换页的地方。

2) 环境不能用在任何列表中，也不能用在列表前面和后面而紧挨着列表。它与列表之间必须有一个空行或分段命令，如 \par。

3) 环境一般放在段落的最前面。如果要将图形或表格置于一个段落中间，那么必须把环境放在有一个自然换行的两个词之间。

4) 多栏排版的页面中不能使用这两个环境。

可以用 wrapfigure 环境按下面的方法：

```
\newcommand{\shouzi}[1]{%
\newlength{\Height}
\settoheight{\Height}{#1}
\intextsep=0pt\columnsep=1pt \noindent
\begin{wrapfigure}{l}{\baselineskip+\Height}
\mbox{\fontsize{1.5\baselineskip+\Height}%
```

```
        {1.5\baselineskip+\Height}\selectfont #1}
    \end{wrapfigure}}
```

定义一个命令 \shouzi，用来使段落的首字放大，使它的高度占两行。例如：

```
\shouzi{水}陆草木之花，可爱者甚蕃。晋陶渊明独爱菊；自李唐来，世人盛爱牡丹；予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，亭亭静植，可远观而不可亵玩焉。予谓菊，花之隐逸者也；牡丹，花之富贵者也；莲，花之君子者也。噫！菊之爱，陶后鲜有闻；莲之爱，同予者何人；牡丹之爱，宜乎众矣。
```

水陆草木之花，可爱者甚蕃。晋陶渊明独爱菊；自李唐来，世人盛爱牡丹；予独爱莲之出淤泥而不染，濯清涟而不妖，中通外直，不蔓不枝，香远益清，亭亭静植，可远观而不可亵玩焉。予谓菊，花之隐逸者也；牡丹，花之富贵者也；莲，花之君子者也。噫！菊之爱，陶后鲜有闻；莲之爱，同予者何人；牡丹之爱，宜乎众矣。

8.8.3 子图形和子表格

有时需要在一个图形环境中排列若干小图形，也可能在一个表格环境中包含几个小表格，这种图形中的图形和表格中的表格分别称为子图形和子表格。subfigure 就是这样一个用来排版子图形和子表格的宏包。每个子图形或子表格都可以有自己的标题，整个图形或表格有一个总标题。要使用这个宏包必须在源文件的导言部分输入命令：

```
\usepackage[选项]{subfigure}
```

其中的选项用来指明如何处理标题，可以取 normal, hang, center, centerlast, nooneline, scriptsize,···, Large, up, it, sl, sc, md, bf, rm, sf, tt。这些选项的意义与 caption2 宏包的选项一样。

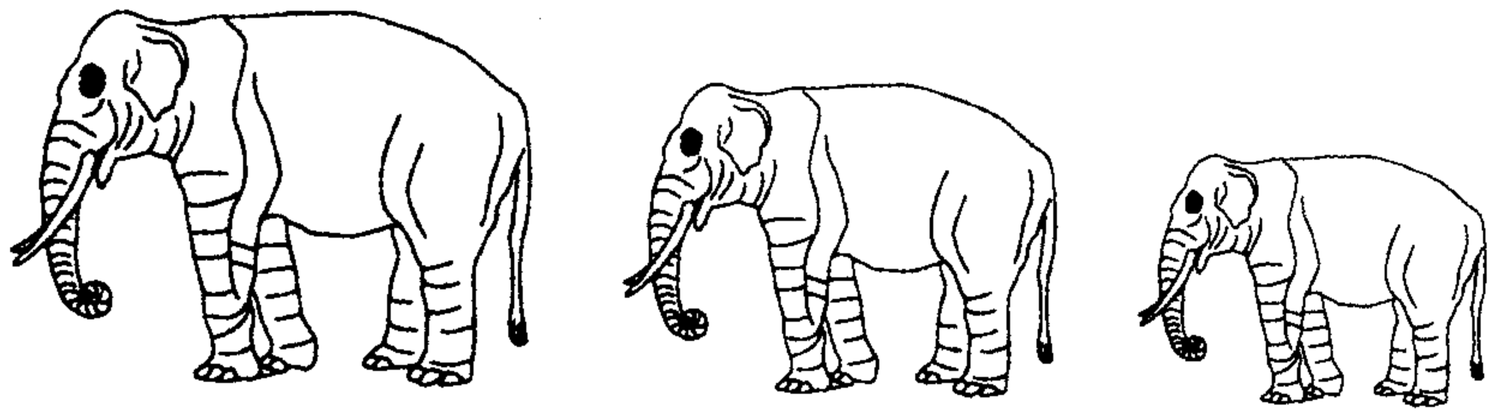
在图形环境和表格环境中我们可以分别用命令

```
\subfigure[子图形标题]{子图形},
\subtable[子表格标题]{子表格}
```

来装载子图形和子表格，如图 8.3 所示。

一般的情况下子图形和子表格标题的标记形式是英文字母外加圆括号，即 (a), (b) 等。用来定义标记形式的命令是 \thesubfigure，指向标记序号的计数器是 subfigure。不管子图形有没有标题，每一个 \subfigure 命令都使计数器 subfigure 的值增 1。如果有必要可以重新定义 \thesubfigure 来获得不同形式的标记。对称地，也可以重新定义 \thesubtable 来获得不同形式的子表格标题的标记。子图形和子表格标题字体由命令 \subcaplabelfont 来定义，子图形和子表格标题字体的大小的默认定义是 \footnotesize, 但可以重新定义命令 \subcapsize 来改变。例如下面的命令：

```
\begin{figure}
\centering
\mbox{\subfigure[Big]{\includegraphics[width=.30\textwidth]{elephant.eps}}\quad
      \subfigure[Medium]{\includegraphics[width=.25\textwidth]{elephant.eps}}\quad
      \subfigure[Small]{\includegraphics[width=.20\textwidth]{elephant.eps}}}
\caption{3 个子图形}\label{fig:subfigure}
\end{figure}
```



(a) Big (b) Medium (c) Small

图 8.3 3 个子图形

```
\renewcommand{\subcaplabelfont}{\bfseries}
\renewcommand{\subcapsize}{\normalsize}
\renewcommand{\thesubfigure}{\thefigure.\arabic{subfigure}}
\makeatletter
\renewcommand{\@thesubfigure}{{\subcaplabelfont\thesubfigure}:\space}
\renewcommand{\p@subfigure}{}
\makeatother
```

将子图形标题字体的大小改为 \normalsize, 并且把标记改为黑体字。标题样式形如 "**2.1**: Big" 其中 2 是整个图形的序号，1 是子图形的序号，Big 是子图形标题内容。另外，subfigure 宏包还提供了下面的命令来控制子图形和子表格的布局：

\subfigtopskip	子图形或子表格的顶部到外层 figure 或 table 环境的起点之间的距离，默认值为 10pt。
\subfigcapskip	子图形或子表格的底部到它们的标题之间的距离，默认值为 10pt。
\subfigbottomskip	子图形或子表格标题的下面留出的空白距离，默认值为 10pt。
\subfigcapmargin	子图形或子表格两边留出的空白距离，默认值为 10pt, 并且只能取正值。

如果要将子图形或子表格的标题也排版到图形或者表格的目录中，那么只需在使用命令 `\listoffigures` 和 `\listoftables` 之前把计数器 `lofdepth` 和 `lotdepth` 的值设置为 2, 也就是使用命令

```
\setcounter{lofdepth}{2}  和  \setouunter{lotdepth}{2}
```

第 9 章　排版中文稿件

在国内用 LaTeX 来排版中文稿件通常有两种方法，一种是用中国科学院计算数学与科学工程计算研究所等单位开发的 CCT 系统，另一种是用 Werner Lemberg 开发的 CJK 宏包。这两种方法各有长处。CCT 系统是通过预处理的办法将含有中文文字的扩展名为 ctx 的源文件转为 TeX 文件，再用 TeX 系统对其进行编译产生 dvi 文件。由于这个 dvi 文件中含有中文信息，必须用 CCT 系统提供的驱动程序进行预览和打印，或者利用 patchdvi 等程序将它转成通常的 dvi 文件，然后再用一般的 dvi 预览程序进行预览和打印。CCT 系统中提供了适合中文稿件规范的文件类型 cctart.cls 和 cctbook.cls，因而用户不需要进行多少设置就能排版出较为规范的文稿。而使用 CJK 宏包来排版中文 LaTeX 稿件就不需要对源文件进行预处理，可以直接用 TeX 系统对其编译，所产生的 dvi 文件可以用一般的 dvi 预览程序进行预览和打印。CCT 系统最初开发出来时是与 emTeX 系统捆绑在一起的，但在科技发展日新月异的今天，emTeX 系统已显得有些陈旧了。由于 emTeX 系统开发者基本上不再对其升级，因而许多 TeX 和 LaTeX 的新功能（如 pdfTeX, LaTeX2Html 等）都不能在 CCT 中使用。虽然最新的 CCT 系统已能脱离 emTeX 而在其他一些 TeX 系统中运行，但还有待进一步发展。而 CJK 是一个宏包并不依赖某个特别的系统，它只是告诉系统如何处理中日韩等亚洲语言文字的信息，因而它不仅能安装到各种 TeX 系统中，而且 TeX 和 LaTeX 的新功能都能很好的得到应用。本书的排版就是使用 LaTeX2e 和 CJK 宏包并辅以其他一些宏包而完成的。在这一章里我们主要阐述如何在 LaTeX2e 中安装和使用 CJK 宏包。

9.1　CJK 宏包的安装

CJK 宏包是一个标准的 LaTeX 宏包，有些新版本的 TeX 系统（如 MiKTeX 2.1）已经包含 CJK 宏包。CJK 宏包本身的安装非常简单，只需将所取得的 CJK 宏包压缩文件中的 texinput 目录下的所有子目录和其中的文件复制到 TeX 系统放置各种宏包的目录中。比如在 MiKTeX 系统中，可以将这些目录和文件放在 `texmf\tex\latex\cjk\` 中。在 emTeX 系统中，可以放在 `emtex\texinput\cjk\` 中。但这只是第一步，因为大多数 TeX 系统都没有安装中文字库，所以要真正能使用 CJK 宏包还必须正确安装和设置中文字库。

9.2 中文字库的安装

CJK 宏包本身不带有任何中文字库，用户必须根据自己的需要安装相应的中文字库。目前，CJK 能支持的中文字库包括：

hbf 字库　这是一种点阵字库。用这种字库生成的 PS 和 PDF 文件在放大时锯齿现象非常明显，打印效果不是很清晰。这是起初没有免费的 Truetype 字库和 Type1 字库时，不得已才选择的字库，因此不建议使用。

Truetype 字库　这是目前 Windows 系统中所使用的一种全真字库。打印效果还可以，但所生成的 PS 和 PDF 文件在放大时仍有锯齿现象。

Type 1 字库　Type 1 字库是一种高质量的矢量字库，有两种格式：PFA 格式（Printer Font Ascii）及 PFB 格式（Printer Font Binary）。用它生成的 PS 和 PDF 文件即使放大到最大倍数也毫无锯齿现象。

以下简要阐述如何安装和使用中文 truetype 字库和 type 1 字库。

9.2.1 使用 Truetype 字库

在 TeX 中使用 Truetype 字库有两种方法，一是直接使用 Truetype 字库，而不需要相应的 pk 字库。二是用 ttf2pk、ttf2tfm 这两个程序。目前只有 pdfTeX 和 pdfLaTeX 可以直接使用 Truetype 字库，dvips 以及大多数 dvi 预览程序都不能直接使用 Truetype 字库。因此我们只介绍如何使用 ttf2pk 和 ttf2tfm 程序。这两个程序是 FreeType 的一部分，MiKTeX 中已安装，但 teTeX 中没有包含，需要自己安装。每种字库都有自己的编码方式，中文 Truetype 字库的编码方式有以下几种：

GB　GB2310—1980 编码的中文字库，国内常用的字库编码形式。

GBK　GB2310 的扩展字符集，是一种包含更多中文字符的字库。

BIG5　BIG5 方式编码的中文字库，是中国台湾及香港等地常用的字库编码形式。

GBt　类似于 GB 编码，但字体是繁体。

9.2.1.1 编辑 ttfonts.map 文件

由于在 TeX 系统中，dvi 预览程序需要 pk 字体来预览 dvi 文件，所以必须让系统能够通过 Truetype 字库自动产生 pk 字体，这就需要编辑 ttfonts.map 文件。在 MiKTeX 中此文件在目录 `texmf\ttf2tfm\base` 里。此文件的大致格式如下：

```
gbst@UGB@       simsun.ttf          Pid = 3  Eid = 1
gbkt@UGB@       simkai.ttf          Pid = 3  Eid = 1
gbwb@EUC@       gwei00b.ttf         Pid = 3  Eid = 3
```

```
gbst@UGB@       simsun.ttf    Slant = 0.25    Pid = 3  Eid = 1
gbkt@UGB@       simkai.ttf    Slant = 0.25    Pid = 3  Eid = 1
gbwb@EUC@       gwei00b.ttf   Slant = 0.25    Pid = 3  Eid = 3
```

其中第 1 列中符号 @ 前面的部分是 pk 字体的主名称，也就是在 CJK 的字体定义文件中的字体名称。两个 @ 中间的部分是 Truetype 字库的子字库定义方式（sub font definition）。每种子字库定义方式都对应着一个以 .sfd 为扩展名的子字库定义文件，比如 UGB.sfd。这些文件也都存放在目录 texmf\ttf2tfm\base 里。由于中文字库包含的字符都很多，TeX 无法直接一次读取，而必须把它分成一些子字库，子字库定义文件就是原先的字库在子字库中的映射关系。与中文有关的子字库定义方式有如下几种：

UGBK 这种定义方式对应于包含 GBK 字符集而且字库内部使用了 Unicode 方式存储的 Truetype 字库。

UGB 这种定义方式对应于包含 GB 字符集而且字库内部使用了 Unicode 方式存储的 Truetype 字库。

UBig5 这种定义方式对应于包含 Big5 字符集而且字库内部使用了 Unicode 方式存储的 Truetype 字库。

EUC 这种定义方式对应于包含 GB 字符集而且字库内部也使用了 GB 方式存储的 Truetype 字库。

Big5 这种定义方式对应于包含 Big5 字符集而且字库内部也使用了 Big5 方式存储的 Truetype 字库。

第 2 列是真实的 Truetype 字库文件名。第 3 列决定是否对字体进行倾斜处理。第 4 列的参数 Pid 的值与所使用的系统平台有关，对于 Windows 系统来说这个值为 3。第 5 列的参数 Eid 的值对应每种字库的存储方式。当 Pid 的值为 3 时，GB 存储方式对应的 Eid 值为 3, Unicode 存储方式对应的 Eid 值为 1, BIG5 存储方式对应的 Eid 值为 4。对于其他系统中所对应的 Pid 值和 Eid 值可查阅 ttf2pk 程序的说明文件。

9.2.1.2 产生 TFM 字体

TFM 字库是编译 TeX 源文件生成 dvi 文件时所必需的。因为 TeX 不能自动从 Truetype 字库生成 TFM 字库，所以需要事先用 ttf2tfm 程序将这些字库造出来。假如有一个以 GB 方式编码并且内部以 Unicode 方式储存的 Truetype 字库，如 Windows 自带的宋体字库 simsun.ttf 就是这类字库，那么可以在命令行中键入如下两条命令：

```
ttf2tfm  simsun.ttf -P 3 -E 1 gbst@UGB@
ttf2tfm  simsun.ttf -S 0.25 -P 3 -E 1 gbstsl@UGB@
```

来产生与这个 Truetype 字库对应的所有直立的和斜体的 TFM 字库（Truetype 字库必须放在 TeX 能搜寻到的目录里）。然后将这些 TFM 字库移到 TeX 系统存放 TFM 字

库的地方，比如放在 texmf\fonts\tfm\chinese\gbst\ 中。注意，上面这两行命令是与 ttfonts.map 文件中相应的行对应的。

9.2.1.3 编辑或修改字体定义文件

TeX 是通过字体定义文件来找寻相应的字体的。如果没有字体定义文件或者定义文件出错，就算已经安装了字体，系统仍会找不到相应的字体。在 LaTeX2e 中字体定义文件以 fd 为扩展名。CJK 的字体定义文件符合 LaTeX2e 的 NFSS 规格，其文件名形式为 C??*name*.fd。这里 *name* 是定义的字体所在的字族名，它至多由 5 个字母组成，两个问号代表数字，C?? 是为 NFSS 所用的对应于字体所在字库的编码方式。下面列出了与中文字库有关的对应：

C00 代表 Big5 字符集的 CJK 默认方式字库。
C01 代表 Big5 字符集的 CJK 的 pmC 方式字库。
C09 代表 Big5 扩展字符集的 CJK 默认方式字库。
C10 代表 GB 字符集的 CJK 默认方式字库。
C11 代表 GB 字符集的 CJK 的 pmC 方式字库。
C19 代表 GBK 字符集的 CJK 默认方式字库。
C20 代表 GBt 字符集的 CJK 默认方式字库。
C21 代表 GBt 字符集的 CJK 的 pmC 方式字库。

比如按照上面两行命令所造出来的 TFM 字库，所对应的字体定义文件名可为 C10st.fd。此文件的内容包含下面 3 个主要命令：

1) \ProvidesFile{C10st.fd}{*release-info*}: 说明本文件避免重复载入。

2) \DeclareFontFamily{C10}{st}{}: 说明本文件以 C10 对应的方式编码且描述 st 字族。

3) \DeclareFontShape{C10}{st}{m}{n}{<-> CJK * gbst}{}: 此命令第 5 个参数较复杂，可参阅 CJK 宏包中的 fonts.doc 说明文件。其中 * 号右边的参数是一组字体文件名的字首，CJK 系统会在其后加上二位数字，成为子字库名称（sub-font name），不论是tfm 文件、pfb 文件还是 pk 文件都与此名一致。其中的 CJK 则是字体大小函数（size function）。CJK 系统定义了数个这样的函数，如：

CJK 相当于一个空格(empty)，意指使用字的大小与 LaTeX 编译的一致。
sCJK 相当于 s，意指使用字的大小与 LaTeX 编译的一致，但取字有误时不警告（silent）。
CJKfixed 相当于 fixed，表示当需要的字的大小在定义范围内时，一律使用指定的字。

sCJKfixed 相当于 sfixed，表示需要的字的大小在定义范围内时，一律使用指定的字，但取字有误时不警告（silent）。

CJKsub 相当于 sub，用其他字体替代。

CJKssub 相当于 ssub，用其他字体替代，但取字有误时不警告（silent）。

下面是一个完整的字体定义文件实例：

```
\def\fileversion{4.3.0}
\def\filedate{1999/06/20}
\ProvidesFile{c10st.fd}[\filedate\space\fileversion]

% simplified Chinese characters
% character set: GB 2312-80
% font encoding: CJK (standard)

\DeclareFontFamily{C10}{st}{}
\DeclareFontShape{C10}{st}{m}{n}{<-> CJK * gbst}{}
\DeclareFontShape{C10}{st}{bx}{n}{<-> CJK * gbht}{}
\DeclareFontShape{C10}{st}{m}{it}{<-> CJK * gbstsl}{}
\DeclareFontShape{C10}{st}{bx}{it}{<-> CJKb * gbstsl}{\CJKbold}
\DeclareFontShape{C10}{st}{m}{sl}{<-> CJK * gbstsl}{}
\DeclareFontShape{C10}{st}{bx}{sl}{<-> CJKb * gbstsl}{\CJKbold}
\endinput
```

从这个例子可看出：st 字族中直立的字体从以 gbst 为首的子字库中取，黑体从以 gbht 为首的子字库中取，两种斜体都从以 gbstsl 为首的子字库中取。由于并非所有的字体都能找到相应的黑体，为此 CJK 通过平移字模后重叠来得到加粗的字体。但这种方法的效果并不是太好，特别是放大后可以看到明显的锯齿痕迹。如果要使用这种办法来得到加粗的字体，就在命令 \DeclareFontShape 后面的最后一对花括号中填入 \CJKbold，否则就空着。

9.2.2 使用 type1 字体

Type 1 字库是一种 PostScript 字库，它是质量很好的矢量字库，用它生成的 PostScript 和 PDF 文件也很漂亮。在专业排版行业，PostScript 字库的应用越来越多，这种字库也逐渐成为 TeX 系统的主要字库。但是目前免费的中文 Type 1 字库很少，而且由于 TeX 系统还同时需要与之相关的 TFM 文件和 AFM 文件，所以能够直接下载的中文 Type 1 字库并不多，只有中文 TeX 网站 http://www.ctex.org/ 中提供了几种这类字库。如果觉得不够用可以利用 gbpfb 程序将自己喜爱的中文 Truetype 字库转为

Type 1 字库，并得到相关的文件。gbpfb 程序是陈向阳在 Freetype 的基础上修改了其中的 ttf2pfb 和 t1asm 而得到的一个工具。它支持 GBK 编码的字库，能生成 CJK 默认方式和 pmC 方式两种 Type 1 字库。不过这个程序只能在 unix 或 linux 系统中使用，若要在 Windows 系统中使用则需要借助于 cygwin 或 djgpp。具体如何产生 Type 1 字库，请参阅相关的程序说明文件。假设已经得到一种字型的中文 Type 1 字库和相应的文件，比如它们包括：

```
gbkst??.pfb    gbkst??.tfm    gbkstsl??.tfm    gbkst??.afm    gbkst.map
```

那么，首先要将所有 gbkst??.pfb 文件复制到 TeX 系统存放 Type 1 字库的地方。在 MiKTeX 系统中，可以放在下面的目录中：

```
texmf\fonts\Type1\chinese\gbkst
```

其次将所有 gbkst*.tfm 文件复制到 TeX 系统存放 TFM 文件的地方。MiKTeX 系统中，可以放在

```
texmf\fonts\tfm\chinese\gbkst
```

目录里。然后将所有 gbkst??.afm 文件复制到 TeX 系统存放 AFM 文件的地方。以 MiKTeX 系统为例，可以放在

```
texmf\fonts\afm\chinese\gbkst
```

目录里。最后将 map 文件 gbkst.map 复制到 dvips 能够搜寻到的地方。一般放在 texmf\dvips 的下级目录里。map 文件告诉系统如何处理 Type 1 字库。它的每一行的格式大致是：

```
gbkst81 GBK-Song81 < gbkst81.pfb
```

如果有许多种中文 Type 1 字库，那么可以将它们的 map 文件合并成一个大的 map 文件，不妨将这个大的 map 文件起名为 CJK.map，要让系统能使用所安装的中文 Type 1 字库，还必须在 dvips 的配置文件 config.ps 里加入下面一行：

```
p +CJK.map
```

注意，这一行的最前面不能是空格。配置文件 config.ps 一般存放在

```
texmf\dvips\config\
```

目录里。另外，如果想让 pdfTeX 也能使用中文 Type 1 字库，还必须在它的配置文件 pdftex.cfg 里加上一行

```
map +CJK.map
```

注意，这一行的最前面也不能是空格。pdfTeX 的配置文件 pdftex.cfg 一般放在

```
texmf\pdftex\config\
```

目录里。

在 MiKTeX 系统中，必须正确配置 ttfont.map 文件才能使 Yap 程序预览带有斜体中文字的 DVI 文件时自动生成所需的 pk 字库。这就必须知道 Type 1 字库是由哪些 Truetype 字库产生的。另外，安装中文 Type 1 字库的同时也必须配置相应的字库定义文件，如果中文 Type 1 字库是 CJK pmC 方式的，那么就需要名如 C11st.fd 的字库定义文件，而当你的中文 Type 1 字库是 CJK 默认方式（GBK 方式）时，就需要名为 C19st.fd 的字库定义文件。编辑字库定义文件的方法已在 9.2.1.3 节里叙述过了。

使用 CJK 宏包，最困难的一步就是要正确安装中文字库和配置相应的文件。如果觉得以上叙述不够详细和明了的话，请参阅其他有关文章。

9.3 CJK 的环境和命令

如果已经正确安装和配置了中文字库，那么就可以开始使用 CJK 宏包来编写包含中文字体的 LaTeX 源文件了。CJK 宏包实际上是可以排版包含中、日、韩在内的好几种亚洲文字的，而且可以混合使用这些文字。这里只介绍与中文字体有关的一些环境和命令。

9.3.1 基本 CJK 环境

使用 CJK 宏包来编写 LaTeX 源文件，首先要在源文件的导言部分用

```
\usepackage{CJK}
```

来调用 CJK 宏包。其次所有中文字符都要包含在环境

```
\begin{CJK}[fontencoding]{encoding}{family}
...
\end{CJK}
```

或者带星号的环境

```
\begin{CJK*}[fontencoding]{encoding}{family}
...
\end{CJK*}
```

之中。这两个环境不仅可以用在源文件的主体部分也可以用在源文件的导言部分。在 LaTeX2e 中，一个完整的使用 CJK 宏包的源文件大致是如下形式：

```
\documentclass{article}
\usepackage{CJK}
\begin{CJK}[fontencoding]{encoding}{family}
... (一些包含中文信息的定义和设置)
\end{CJK}
```

```
\begin{document}
\begin{CJK}[fontencoding]{encoding}{family}
··· (包含中文字体的文件的主要内容)
\end{CJK}
\end{document}
```

环境中的参数 encoding 表示字体的编码方式，与中文有关的值有：

Bg5　　Bg5+　　GB　　GBK　　GBt

选项 fontencoding 表示是按何种方式来安排子字库的，对中文来说只有两种方式，即默认方式和 pmC 方式。如果是默认方式，就不必给出这个选项。如果是 pmC 方式，这个选项的值应取 pmC。参数 family 表示所用字体属于哪个字族。注意每种字族都对应着一个字体定义文件。如果已安装了宋体、仿宋体、黑体和楷体这四种基本字族，假设对应的字体定义文件分别为 C10st.fd、C10fs.fd、C10ht.fd 和 C10kt.fd，那么 family 的值就可以取 st、fs、ht 或者 kt。参数 family 的值为 st 时，CJK 环境中的主要字体就是宋体的。如果不给出 family 的值，系统会自动用 CJK.enc 中定义的 song 来替代。当然在环境中还可以用其他的命令来选择不同字族和不同编码的部分文字。

习惯上中文文字的排版都是一个紧接着一个的，两个相邻文字之间不留空格。但是按照 TeX 的规则，换行时则会插入一个空格。因此，使用 CJK 环境来排版中文文字时应注意：如果一行的最后一个字符是中文而且下一行的第一个字符也是中文，那么这一行的行尾就应该加一个 % 以避免两个中文字符之间出现多余的空白。如果是大段大段的中文，那么这种方法就显得很笨拙，此时，应使用 CJK* 环境。在这个带星号的环境中，每个中文字符不管其后将会出现什么文字都会毫不客气地吞没紧接其后的未受保护的空白，因此，在换行时两个中文字符之间也不会出现多余的空白。然而我们又希望中文字符与其他字符（如英文字母或数学公式）之间有一个空格，为解决此问题，CJK 宏包提供了命令：

```
\CJKtilde
```

它修改了其后的波浪符 ~ 原先的意义，使得这个符号不再产生一个不可分割的空格，而是产生一个可分割的通常空格。于是，在 CJK* 环境中我们需使用 \CJKtilde 和波浪符 ~ 来避免某些必要的空格被吞吃。例如：

```
许多科技工作者都喜欢用~LaTeX~或者
~TeX~来排版论文。
```

许多科技工作者都喜欢用 LaTeX 或者 TeX 来
排版论文。

许多科技工作者都喜欢用 LaTeX 或者 TeX 来排版论文。

许多科技工作者都喜欢用LaTeX 或者TeX 来排版论文。

上面第二个例子中 LaTeX 和 TeX 前面的空格被吞吃了，这使得英文字母与中文字符之间显得不很协调，而第一个例子就没有这个问题。一篇中文稿件中若出现大量英文字符或数学公式，那就要使用很多的波浪符，这使源文件的输入变得较慢，但为了得到美观的排版效果，目前也只好如此了。

在排版中文稿件时，通常也都希望目录名“Contents”、摘要名“Abstract”等，都能自动换成中文的“目录”、“摘要”等，此时可以使用命令：

```
\CJKcaption{GB}
```

这条命令将加载名为 GB.cpx 的文件，而这个文件里一般预先重新定义了 LaTeX 中的一些名称，使其符合中文习惯。

9.3.2 改变 CJK 字体的命令

为了能同时使用不同编码的文字，以及同一种编码的不同字族，CJK 宏包中提供了其他一些命令。

```
\CJKenc{encoding}
```

这条命令将用以 encoding 为编码的 CJK 字符来排版其后的文字。比如在以 GB 为编码的 CJK 环境中使用命令 \CJKenc{GBK} 时，其后的文字将以 GBK 为编码。

```
\CJKfontenc{encoding}{fontencoding}
```

这条命令将其后面的以 encoding 为编码的 CJK 字符的子字库安排方式改成编码为 fontencoding 的子字库安排方式。比如 \CJKfontenc{GB}{pmC} 将用 GB 编码的 pmC 方式的字体来排版其后的文字。上面这两条命令都不改变文字的字族属性，因此使用时应注意是否存在相应的字体定义文件。比如在环境

```
\begin{CJK}{GB}{st} ··· \end{CJK}
```

中使用 \CJKfontenc{GB}{pmC} 时，系统中应同时存在字体定义文件 C10st.fd 和 C11st.fd。另外，上面这条命令可以在 CJK 和 CJK* 这两个环境之外使用。

```
\CJKfamily{family}
```

这是一个很常用的改变字体的命令，它将用以 family 为字族的字体来排版后面的文字而不管后面的文字是什么编码的。例如：

前面这些文字都是宋体文字。\CJKfamily{xk}
从现在开始我们使用行楷字体。

前面这些文字都是宋体文字。 *从现在开始我们使用行楷字体。*

```
\CJKencfamily[<fontencoding>]{<encoding>}{<family>}
```

这条命令将把其后的文字以 <encoding> 为编码，同时以 fontencoding 为子字库安排方式的 CJK 字符的字族改为 family，当然系统中必须存在与字族 family 相应的字体定义文件。这条命令也可以用于 CJK 和 CJK* 这两个环境之外。

9.3.3 与空格有关的命令

如前所述，CJK 环境是按通常的方式来处理空格的，也就是说中文字符后面的空格将会保留。而 CJK* 环境中的中文字符会吞吃其后的空格。在 CJK* 环境中使用命令

```
\CJKspace
```

将把其后的文字切换到 CJK 环境的模式。而在 CJK 环境中使用命令

```
\CJKnospace
```

则将把它后面的文字切换成 CJK* 环境的模式。如果在 CJK* 环境中使用了命令 \CJKtilde，则其后的波浪符 ~ 的意义已被改变。此时我们可以使用命令

```
\nbs
```

来替代波浪符 ~ 原先的意义。因此，Theorem\nbs1 中的 Theorem 和 1 之间会有一个空格，而且它们不会被排版到不同的行。即使在 CJK* 环境中使用了命令 \CJKtilde，我们还可以用下面的命令

```
\standandtilde
```

将波浪符 ~ 回归到原先的定义。如果因为使用中文字符而出现了 overfull \hbox 的警告信息，有时可以通过增大

```
\CJKglue
```

的值来消除。\CJKglue 原先的定义是：

```
\newcommand{\CJKglue}{\hskip 0pt plus 0.08\baselineskip}
```

它是一个空白长度，将被插入到两个相邻中文字符之间。利用这条命令我们可以定义一个新命令用来改变中文字符之间的字距。如

```
\newcommand{\ziju}[1][0]{%
\newlength\CJKonespace%
\newlength\CJKtwospaces%
\settowidth\CJKonespace{\CJKchar{"0A1}{"0A1}}%
\settowidth\CJKtwospaces{\CJKchar{"0A1}{"0A1}%
\hspace{#1\CJKonespace}\hspace{#1\CJKonespace}\CJKchar{"0A1}{"0A1}}%
\parindent=\CJKtwospaces\renewcommand{\CJKglue}{%
\settowidth\CJKonespace{\CJKchar{"0A1}{"0A1}}%
\hskip #1\CJKonespace plus 0.08\baselineskip}}
```

定义了命令 \ziju，它有一个选项用来指明其后中文文字的字距是当前字宽的多少倍，选项的默认值是 0，即不改变字距。注意，这样定义的命令也改变了缩格。例如：

```
\ziju[0.5]在这个例子中，文字间的距离是
当前字宽的一半。\par \ziju
这一行文字的字距又调整到默认的状态。
```

在这个例子中，文字间的距离是当前字宽的一半。

这一行文字的字距又调整到默认的状态。

通常在排版文稿时，第一个段落是不缩格的，其后的段落虽然有缩格，但这个缩格并不符合中文的习惯，因为中文段落要求所有段落都有两个字符宽度的缩格。新版的 CJK 宏包提供了下面的命令：

\CJKindent

它指示其后的所有段落都有两个字符宽度的缩格，来满足中文排版的要求。在 CJK 4.50 以前的版本中没有这条命令，不过用户可以按下面的方法自己定义这条命令：

```
\newlength\CJKtwospaces
\def\CJKindent{%
\settowidth\CJKtwospaces{\CJKchar{"0A1}{"0A1}\CJKchar{"0A1}{"0A1}}%
\parindent\CJKtwospaces}
```

在前面定义字距命令 \ziju 时，我们同时将两倍字距加入缩格之中。因此使用 \ziju 与使用 \CJKindent 是一样的。还可以利用 \CJKglue 按下述方法

```
\newcommand{\CJKfill}[1]{{%
\renewcommand{\CJKglue}{\hspace{\fill}}\par\noindent #1}}
```

定义命令 \CJKfill，此命令将不满一行的中文文字拉伸使之占满一行，例如：

```
\CJKfill{不满一行的文字}
```

不　满　一　行　的　文　字

9.3.4 CJK 中的其他宏包

在 CJK 宏包中还附带有其他几个有用的宏包，它们是：

CJKnumb 此宏包定义了将阿拉伯数字转为中文数字的命令：

```
\CJKnumber{阿拉伯数字} 和 \CJKdigits{阿拉伯数字}
```

例如，输入命令 \CJKnumber{2003} 得到的是：二千零三。输入命令 \CJKdigits{2003} 得到的是：二〇〇三。

CJKulem 此宏包改进了 ulem 使得更好地在中文之下排版下划线。例如，输入 \uline{文字的下划线}，得到：文字的下划线。

CJKvert 此宏包重新定义了 \CJKsymbol 使得用 \CJKbold 方式构造加粗字体时，是在垂直的方向上而不是在水平的方向上，移动并多打印几次文字，从而得到加粗的字体。这个宏包一般不需要用。

pinyin 此宏包用来输入中文拼音。例如输入 \zhong{1} \guo{2} 将排印出“zhōng guó”。

ruby 此宏包用于在文字上方加拼音或其他符号，例如输入 \ruby{中}{\zhong{1}} \ruby{国}{\guo{2}} 得到：zhōng guó 中 国。

利用命令 \CJKnumber 和 \CJKdigits 可以定义两条类似于命令 \today 但符合中文习惯的日期命令：

```
\newcommand\chntoday{%
   \number\year\,年\,\number\month\,月\,\number\day\,日}
\newcommand\CHNtoday{%
   \CJKdigits{\number\year}年\CJKnumber{\number\month}月%
      \CJKnumber{\number\day}日}
```

\today\\	April 14, 2004
\chntoday \\	2004年4月14日
\CHNtoday	二〇〇四年四月十四日

9.3.5 中文竖排

从目前的使用来看，我们很少需要对通篇文稿进行竖排，然而在一些特殊场合可能需要将某个版块进行竖排。CJK 宏包中并没有提供对文字进行竖排的命令，不过我们可以通过使用 graphicx 宏包中 \rotatebox 命令将中文文字逆时针旋转 90°，从而对中文稿件进行竖排。当然，还必须在源文件中修订纸张的尺寸定义，并且打印时要选择横式（landscape）打印方式。若在小页环境中将中文字体逆时针旋转 90°，然后再将整个小

页顺时针旋转 90°，就可以将页面中的部分文字竖排。下面用

```
\newcommand{\verticle}{%
    \renewcommand{\CJKsymbol}[1]{%
    \setbox0=\hbox{\symbol{##1}}%
    \newcommand{\POS}{}%
    \ifthenelse{\lengthtest{\ht0<.39\wd0}}%
            {\renewcommand{\POS}{c}}{\renewcommand{\POS}{r}}%
    \makebox[\wd0][\POS]{\rotatebox[origin=lB]{90}{\symbol{##1}}}%
    \ifCJK@bold@%
    \hbox to \CJKboldshift{\hss\makebox[\wd0][\POS]{%
        \rotatebox[origin=lB]{90}{\symbol{##1}}}}%
    \hbox to \CJKboldshift{\hss\makebox[\wd0][\POS]{%
        \rotatebox[origin=lB]{90}{\symbol{##1}}}}%
    \fi}}
```

来定义命令 \verticle，它将其后的中文字体逆时针旋转 90°。下面再用

```
\newsavebox{\saverotate}%
\newcommand{\shupai}[2][\textheight]{%
   \savebox{\saverotate}{\parbox[t]{#1}{\verticle #2}}
   \hfill\rotatebox[origin=lt]{-90}{\usebox{\saverotate}}}
```

定义命令 \shupai[*height*]{一些文字}，它将一些文字放在高为 *height* 的小页中，并进行竖排。例如：

```
\shupai[3cm]{\centering
{\Large 黄鹤楼}\\[2mm]
{\large 崔颢}\\[2mm]
昔人已乘白云去，\\ 此地空余黄鹤楼。\\
黄鹤一去不复返，\\ 白云千载空悠悠。\\
晴川历历汉阳树，\\ 芳草萋萋鹦鹉洲。\\
日暮乡关何处是，\\ 烟波江上使人愁。}
```

黄鹤楼
崔颢
昔人已乘白云去，
此地空余黄鹤楼。
黄鹤一去不复返，
白云千载空悠悠。
晴川历历汉阳树，
芳草萋萋鹦鹉洲。
日暮乡关何处是，
烟波江上使人愁。

要注意两点：一是用这种方法进行竖排时，所产生的 dvi 文件中包含 graphicx 宏包的信息，因而某些 dvi 预览程序（如 MiKTeX 中的 Yap）可能不能正确显示，但可以用 dvips 程序将 dvi 文件转换成 PS 文件，然后用 gsview 程序来预览和打印。其次不同版本的字体设计的文字（特别是某些标点符号）的高度可能不同，因此竖排时有些标题符号的位置可能不太合适。在命令 \verticle 的定义中使用了 ifthen 宏包的命令 \ifthenelse 就是为了调整标点的位置。

9.3.6 中文字体的大小

由于 CJK 宏包只是提供了一种使 LaTeX 系统能识别 CJK 文字的方式，它并没有提供改变字体大小的命令。CJK 字体的大小是随着英文字体一起改变的，因而可以用 \small、\large 等命令来选择字体的大小。一般情况下，也没有必要再提供其他的命令。不过国内出版界对中文字体的大小有一套特有的要求，即，中文字体的大小一般在初号至八号之间。出版社有时也会要求作者用五号或者小四号作为排版正文的主要尺寸。那么如何把 Word 及 WPS 中使用的字号尺寸移植到 LaTeX 中呢？由于 Word 中的初号对应 42bp，···，八号对应 5bp。因此可以用下面的设置：

```
\newcommand{\CJKfontsize}[4]{%
        \fontsize{#1}{#2 plus#3 minus #4}\selectfont}
\newcommand\zihao[1]{%
\ifthenelse{\equal{#1}{0}}{%
        \CJKfontsize{42bp}{50.4pt}{.5pt}{.3pt}}{}%
\ifthenelse{\equal{#1}{-0}}{%
        \CJKfontsize{36bp}{43.2pt}{.5pt}{.3pt}}{}%
\ifthenelse{\equal{#1}{1}}{%
        \CJKfontsize{26bp}{31.2pt}{.5pt}{.3pt}}{}%
\ifthenelse{\equal{#1}{-1}}{%
        \CJKfontsize{24bp}{28.8pt}{.5pt}{.3pt}}{}%
\ifthenelse{\equal{#1}{2}}{%
        \CJKfontsize{22bp}{26.4pt}{.5pt}{.3pt}}{}%
\ifthenelse{\equal{#1}{-2}}{%
        \CJKfontsize{18bp}{21.6pt}{.3pt}{.2pt}}{}%
\ifthenelse{\equal{#1}{3}}{%
        \CJKfontsize{16bp}{19.3pt}{.3pt}{.2pt}}{}%
\ifthenelse{\equal{#1}{-3}}{%
        \CJKfontsize{15bp}{18pt}{.3pt}{.2pt}}{}%
\ifthenelse{\equal{#1}{4}}{%
        \CJKfontsize{14bp}{16.8pt}{.3pt}{.2pt}}{}%
\ifthenelse{\equal{#1}{-4}}{%
        \CJKfontsize{12bp}{14.4pt}{.3pt}{.2pt}}{}%
\ifthenelse{\equal{#1}{5}}{%
        \CJKfontsize{10.5bp}{12.6pt}{.3pt}{.2pt}}{}%
\ifthenelse{\equal{#1}{-5}}{%
        \CJKfontsize{9bp}{10.8pt}{.2pt}{.1pt}}{}%
\ifthenelse{\equal{#1}{6}}{%
        \CJKfontsize{7.5bp}{9pt}{.2pt}{.1pt}}{}%
\ifthenelse{\equal{#1}{-6}}{%
```

```
        \CJKfontsize{6.5bp}{7.8pt}{.2pt}{.1pt}}{}%
\ifthenelse{\equal{#1}{7}}{%
        \CJKfontsize{5.5bp}{6.6pt}{.1pt}{.1pt}}{}%
\ifthenelse{\equal{#1}{8}}{%
        \CJKfontsize{5bp}{6pt}{.1pt}{.1pt}}{}}
```

来定义命令 \zihao{n}，它将其后的文字改为 n 号字。这里 n 可以取 $0, -0, 1, -1, 2, -2, 3, -3, 4, -4, 5, -5, 6, -6, 7, 8$ 这些数字，分别表示初号、小初号，一号，小一号，……八号字体。注意，在上面的定义中使用了 ifthen 宏包以及 LaTeX2e 中选择字体尺寸的命令 \fontsize。现在就可以使用命令 \zihao 了：

```
\zihao{5}这是五号字。
\zihao{-4}这是小四号字。\\
\zihao{3}这是三号字。
```

这是五号字。这是小四号字。

这是三号字。

需要注意的是：当采用上面字号的定义时，系统可能会抱怨找不到相应尺寸的某些字体。这是由于TeX 系统对字体是按照 5pt, 6pt, 7pt, 8pt, 9pt, 10pt, $\cdots$ 这样的尺寸来设置的，这与中文字号的尺寸不匹配，因此若所使用的字库不是可以任意伸缩的（如 Type 1 字库），系统会用相近的尺寸来替代。另外还要说明一点：目前在 LaTeX 中对中文字号的定义还没有一个严格的统一标准。

参 考 文 献

[1] David J Buerger. LaTeX for Scientists and Engineers. McGraw-Hill Publishing Company, 1990

[2] Donald E Knuth. The TeXbook. Addison-Wesley, 1986

[3] George Grätzer. Math Into TeX. Birkhäuser, 1993

[4] Helmut Kopka, Patrick W Daly. A Guide to LaTeX–Document Preparation for Beginners and Advanced Users. Addison-Wesley, 1993

[5] Keith Reckdahl. Using Imported Graphics In LaTeX 2_ε. 可在 CTAN 中下载

[6] Leslie Lamport. LaTeX–A Document Preparation System–User's Guide and Reference Manual. Addison-Wesley, 1985

[7] Michel Goossens, Frank Mittelbach, Alexander Samarin. The LaTeX Companion. Addison-Wesley, New York, 1994

[8] Michel Goossens, Sebastian Rahtz, Frank Mittelbach. The LaTeX Graphics Companion. Addison-Wesley, New York, 1997

[9] Tobias Oetiker. The Not So Short Introduction to LaTeX 2_ε. 可在 CTAN 中下载

[10] Victor Eijkhout. TeX By Topic, A TeXnician's Reference. Addison-Wesley, 1991. 电子版可在 CTAN 中可免费获得

[11] 吴凌云，王磊. 中文 LaTeX 扩展安装指南. 可以在 `http://www.ctex.org/` 中下载

[12] 陈志杰，赵书钦，万福永. LaTeX 入门与提高. 北京：高等教育出版社，2002

[13] 邓建松，彭冉冉，陈长松. LaTeX 2_ε 科技排版指南. 北京：科学出版社，2001

[14] 丁卫星，赖天树. LaTeX 实用教程. 合肥：中国科学技术大学出版社，1993

索　引